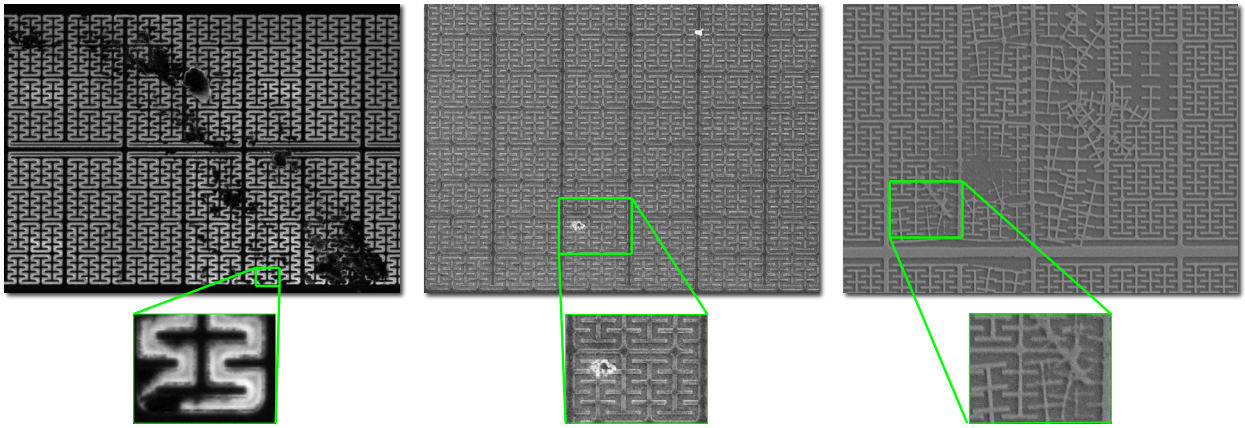
CHALMERS
UNIVERSITY OF TECHNOLOGY



Detektion av defekter i nanostrukturer med maskininlärning

En lösning med Autoencoders och Unsupervised Learning

Kandidatarbete i mikroteknologi och nanovetenskap

ERIC CARLSSON
OLOF CRONQUIST
OSCAR ERIKSSON
MATTIAS ULMESTRAND

INSTITUTIONEN FÖR MIKROTEKNOLOGI OCH NANOVETENSKAP

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2020
www.chalmers.se

KANDIDATARBETE MCCX02-20-06

Detektion av defekter i nanostrukturer med maskininlärning

En lösning med Autoencoders och Unsupervised Learning

ERIC CARLSSON
OLOF CRONQUIST
OSCAR ERIKSSON
MATTIAS ULMESTRAND

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för mikroteknologi och nanovetenskap
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2020

Detektion av defekter i nanostrukturer med maskininlärning
En lösning med Autoencoders och Unsupervised Learning
ERIC CARLSSON
OLOF CRONQUIST
OSCAR ERIKSSON
MATTIAS ULMESTRAND

Utbildning: Teknisk fysik

Handledare: Niclas Lindvall och Sumedh Mahashabde, Institutionen för mikroteknologi och nanovetenskap
Examinator: Vessen Vassilev, Institutionen för mikroteknologi och nanovetenskap

Kandidatarbete MCCX02-20-06
Institutionen för mikroteknologi och nanovetenskap
Chalmers tekniska högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Omslag: Exempel på defekter som förekommer i mikrovågledarkretsar på MC2

Institutionen för mikroteknologi och nanovetenskap
Göteborg, Sverige 2020

Detektion av defekter i nanostrukturer med maskininlärning
En lösning med Autoencoders och Unsupervised Learning
ERIC CARLSSON
OLOF CRONQUIST
OSCAR ERIKSSON
MATTIAS ULMESTRAND
Institutionen för mikroteknologi och nanovetenskap
Chalmers tekniska högskola

Abstract

This study aims to investigate the possibilities of using Machine Learning for detecting fabrication defects in nanostructures. The structures investigated are microwave circuits produced at the Department of microtechnology and nanoscience (MC2) at Chalmers University of Technology. Firstly, a particularly defective circuit was investigated with Logistic Regression, Dense Neural Networks, Convolutional Neural Networks as well a Transfer Learning based method with ResNetV2 and Principal Component Analysis. The distribution of defective and non-defective circuits was large and balanced enough to achieve 100 % correct classification of the sections with almost all models. Two more realistically defective circuits were further investigated, where the defective sections were widely underrepresented. However, a Convolutional Autoencoder (CAE), trained with either Supervised or Unsupervised Learning, was largely successful in separating defective sections from non-defective ones with a clear boundary based on the reconstruction error provided by the CAE. Furthermore, the CAE was in many cases able to locate the exact positions of defects by marking the areas of maximum reconstruction error, and also flag defective sections that previously was unsuccessful with manual inspection. The circuit sections used for the CAE were automatically sampled from larger images of circuits. After being sampled from the larger images of circuits, the sections were only downsampled and not preprocessed in any other way. The success of the unsupervised approach is the main achievement of the study, as all that is needed to train the model is completely uninvestigated images. The approach is estimated to save several days of manually inspecting whole circuits for defects.

Keywords: Machine Learning, Unsupervised Learning, Convolutional Autoencoder, nanofabrication, microwave circuit

Förord

Först och främst vill vi uttrycka vår tacksamhet till Niclas Lindvall och Sumedh Mahashabde för att ha gett oss möjligheten att skriva detta kandidatarbete. De har varit utmärkta handledare och en källa till en mycket trevlig och utvecklande miljö att skriva vårt kandidatarbete i. Vidare har Niclas och Sumedh gett oss en god inblick i att arbeta med nanoteknologi och maskininlärning på Chalmers, och har fått oss att känna oss inkluderade och välkomna i verksamheten på MC2. Vi vill dessutom tacka för alla trevliga fikatillfällen, inklusive de utsökta semlorna med personalen på MC2 på fettisdagen.

Vidare vill vi tacka Elsebeth Schröder och C3SE på Chalmers för att ha gett oss tillgång till PC-kluster för beräkningskraft, vilket har kraftigt accelererat träningen av de neurala nätverken och därigenom även bidragit till resultaten i kandidatarbetet.

Författarna, Göteborg, maj 2020

Bild: Författarna med handledaren Niclas utanför renrummet i MC2. Bilden fångades av handledaren Sumedh.

Förkortningslista

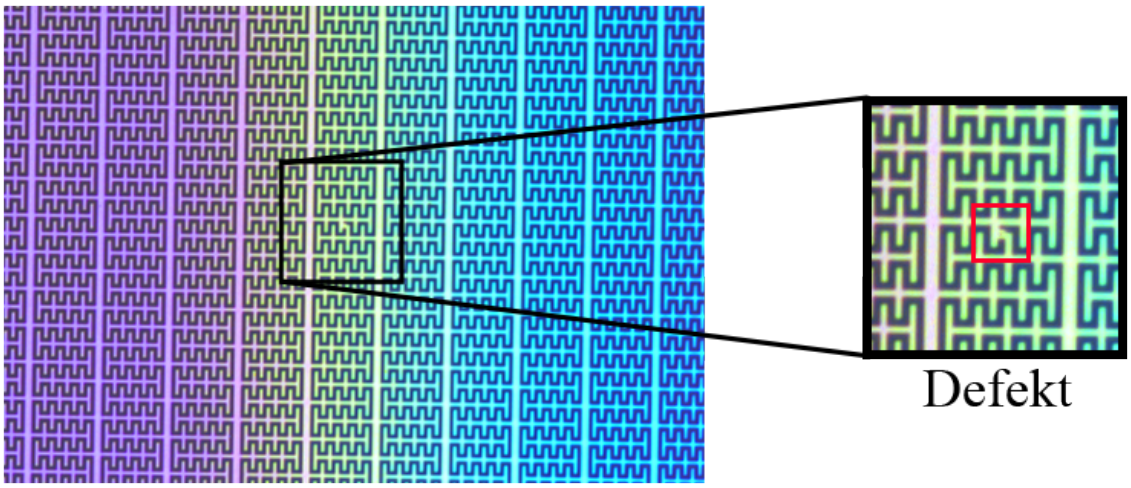
- AE – Autoencoder
- AD – Anomaly Detection
- ANN – Artificial Neural Network
- BCE – Binary Cross-Entropy
- CAE – Convolutional Autoencoder
- CNN – Convolutional Neural Network
- DB – Decision Boundary
- DNN – Dense Neural Network
- GAP – Global Average Pooling
- LR – Logistisk regression
- ML – Machine Learning
- MSE – Mean Squared Error
- MMSEp – Maximum Mean Squared Error, dimension p
- PCA – Principal Component Analysis
- ReLU – Rectified Linear Unit
- SEM – Svepelektronmikroskop
- SVM – Support Vector Machine
- TL – Transfer learning
- VAE – Variational Autoencoder

Innehåll

1 Inledning

Användning av supraledare som designmaterial i mikrovågskretsar har gett upphov till ett helt nytt kunskapsområde inom kretsar för kvantelektrodynamik och kvantberäkningar. Tekniken utnyttjar mikrovågsfotoner som bärare av kvantinformation vilket kräver mikrovågledare i mikrostorlek. På Chalmers institution Mikroteknologi och nanovetenskap (MC2) sker forskning inom området och där tillverkas ledare i ett fraktalmönster för att uppnå fördelaktiga elektriska egenskaper i kretsarna [1]. Ibland uppstår defekter i ledaren vid tillverkning, se Figur 1. I värsta fall innebär en defekt att en kortslutning har skett och att kretsen ej går att använda. De ledande delarna har en bredd på 250 – 500 nanometer, därför används ofta elektronmikroskop för att kunna fånga ledarna med god upplösning. Bilderna inspekteras sedan manuellt för att hitta eventuella kortslutningar. Kortslutningar kan därefter repareras med en fokuserad jonstråle.

Att inspektera mikroskopibilderna manuellt är ett tidskrävande jobb som potentiellt kan effektiviseras med hjälp av maskininlärning. Det skulle då innebära att kortslutningarna automatiskt detekteras genom bildklassificering och tidsåtgången att identifiera defekterna för att sedan kunna reparera ledaren kraftigt kan reduceras. Ännu en fördel är att en maskininlärningslösning kan anpassas för att kunna användas på andra problem inom nanoteknik där feldetektion är nödvändigt. Maskininlärning applicerat för nanotillverkning är vidare relativt outforskat. Därför är ett av målen med denna studie att utveckla kunskap om tillämpning av maskininlärning för att förbättra nanotillverkningprocesser.



Figur 1.1: En tillverkningsdefekt på en mikrovågledarkrets. I detta fall är kretsen kortsluten. Notera att bilden är tagen med optiskt mikroskop. Ofta används svepelektronmikroskop (SEM) för att erhålla högre upplösning på bilderna.

1.1 Syfte

Syftet med denna rapport är att utveckla maskininlärningsalgoritmer som detekterar och markerar fabrikationsdefekter i fraktalformade mikrovågledarkretsar.

2 Teori

För att förstå innehållet i rapporten krävs viss kunskap om uppbyggnaden av maskininlärningsmodeller, speciellt artificiella neurala nätverk (ANN). Studien fokuserar på varianter av ANN, främst Autoencoders (AE), se avsnitt 2.5. Även enklare modeller behandlas, exempelvis Support Vector Machine (SVM) och logistisk regression (LR).

2.1 Maskininlärning

Ett stort problem med att manuellt inspektera mikrovågledarkretsarna är att många stora bilder måste undersökas noggrant, vilket kan vara tröttsamt och tidskrävande. Inspektionen av kretsar lämpar sig därför bättre för att automatisera med algoritmer. En metod som är ett känt verktyg för att klassificera defekter är *Machine Learning* (ML), eller maskininlärning [2].

Maskininlärning är en gren av artificiell intelligens och kan beskrivas som läran om algoritmer som datorer använder sig av utan specifik instruktion. ML-algoritmer förlitar sig på att finna mönster genom att "läras" på vanligtvis stora datamängder, för att efter inlärning kunna dra slutsatser om liknande data. En stor fördel med ML är även att en ML-algoritm lätt generaliseras till andra problem av liknande art genom små justeringar. Omarbetning av en icke-inlärningsbaserad modell kräver typiskt betydligt större insatser [3].

För varje maskininlärningsmetod finns en uppsättning parametrar som måste bestämmas på förhand. Parametrarna benämns hyperparametrar och innefattar bland annat val av modeller och egenskapsdetektion (feature detection). Antalet hyperparametrar och vilka hyperparametrar som existerar varierar mellan olika metoder och kan ha en varierande grad av inverkan på resultat, tidsåtgång och träffsäkerhet. Indatan till maskininlärningsmetoder kallas för egenskaper (features) och valet av indata är i sig en hyperparameter. Ett enkelt exempel på en egenskap vid analys av bilder är färgdjupet av en pixel i en bild.

Att identifiera defekter i mikrovågledarkretsar kan formuleras som ett binärt klassifikationsproblem. Uppgiften blir att skilja på två klasser – defekt eller ickedefekt krets.

2.2 Logistisk regression

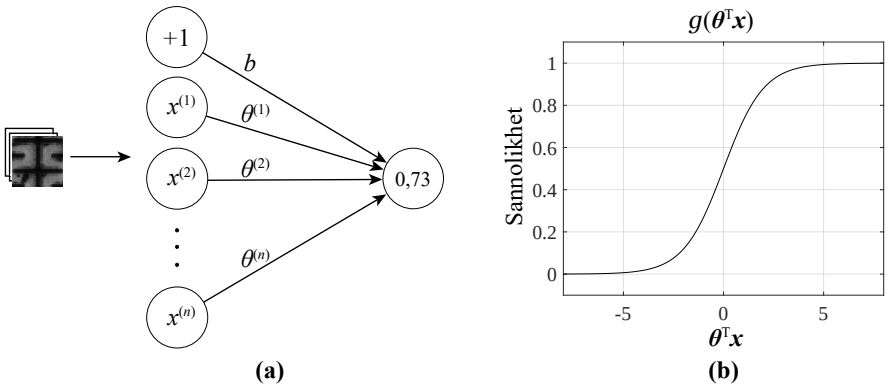
Detta delavsnitt utgår från C. M. Bishop [4] samt A. Ng [5] (föreläsning 6.1-6.7). Logistisk regression är en av de enklaste modellerna inom inlärningsbaserade klassifikationsalgoritmer. Målet med logistisk regression tillämpad för binär klassifikation är att ge en sannolikhet mellan 0 och 1, där 0 representerar att modellen är helt säker på att kretssektionen inte är defekt, och 1 representerar att sektionen helt säkert är defekt.

Indatan som analyseras med logistisk regression finns vanligen lagrad som reella tal i en tensor $\mathbf{x}$. Tensorn $\mathbf{x}$ önskas mappas med hjälp av vikterna i vektorn $\boldsymbol{\theta}$ till vektorn $\mathbf{y}$. Mappningen sker via den så kallade hypotesfunktionen $h_{\boldsymbol{\theta}}$,

$$h_\theta : \mathbf{x} \mapsto \mathbf{y}.$$

Vektorn $\mathbf{y}$ är i binär klassifikation en samling värden som antingen antar värdet 0 eller 1. Värdet kallas för elementets annotering och representerar ickedefekt respektive defekt kretssektion. $\mathbf{x}$ kan exempelvis vara en samling av 2000 bilder som vardera består av 50×50 pixlar. $\mathbf{x}$ har då dimensionen $2000 \times 50 \times 50$. Hypotesfunktionen mappar $\mathbf{x}$ till $\mathbf{y}$ genom den inre produkten av $\mathbf{x}$ och $\boldsymbol{\theta}$ till ett värde mellan 0 och 1. Mappningen görs med funktionen $g(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x})$. Funktionen $g(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x})$ kallas för en aktiveringsfunktion. Värdet som ges av aktiveringsfunktionen kan tolkas som sannolikheten att kretsen är defekt. Aktiveringsfunktionen ges inom LR av den så kallade logistiska funktionen, eller *sigmoid*:

$$g(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}}}. \tag{2.1}$$



Figur 2.1: **(a)** är en modellrepresentation av logistisk regression. Bilder på kretssektioner tolkas genom att pixelintensiteten (flyttal mellan 0 – 1 i gråskala) plattas ned till en vektor och viktas med parametrarna $\theta^{(i)}$ via $g(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x})$. I detta fall finns en defekt på kretssektionen, som kan tänkas ge exempelvärdet 0,73 illustrerat i figuren. Noden med "+1" ger utdata b, vilken benämns biasterm. Biastermen kan behövas för att skifta slutvärdet för att få korrekt förutspåelse. Aktiveringsfunktionen $g(\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}) = (1 + \exp(-\boldsymbol{\theta}^{\mathrm{T}}\mathbf{x}))^{-1}$ visas i **(b)**.

För att kvantifiera felet i förutspåelsen som ges av hypotesen måste kostnadsfunktionen $J(\boldsymbol{\theta})$ definieras,

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}), \tag{2.2}$$

där m är antal datapunkter i $\mathbf{x}$, som i exemplet med 2000 bilder om 50×50 pixlar vardera är antal bilder, $m = 2000$. För att träna parametervektorn $\boldsymbol{\theta}$ önskas kostnadsfunktionen minimeras. Själva minimeringen av $J(\boldsymbol{\theta})$ är ett av huvudmålen i maskininlärning, eftersom det är ett mått på hur korrekt en förutspåelse är. Vanligtvis slumpas initialt parametrarna $\boldsymbol{\theta}$. Eftersom parametrarna då inte är anpassade till problemet får $J(\boldsymbol{\theta})$ typiskt ett högt värde. Funktionen "Cost" är i logistisk regression definierad som

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{om } y = 1 \\ -\log(1 - h_\theta(x)) & \text{om } y = 0 \end{cases},$$

vilket kan skrivas om som

$$\mathrm{Cost}(h_\theta(x), y) = -y\log(h_\theta(x)) - (1 - y)\log(1 - h_\theta(x)). \tag{2.3}$$

Funktionen $\text{Cost}(h_\theta(x), y)$ är i logistisk regression konvex, vilket innebär att ett globalt minimum går att finna. Genom exempelvis *Gradient Descent* eller andra optimeringsalgoritmer minimeras $J(\boldsymbol{\theta})$ för varje iteration av träning. Med rätt implementation blir modellen bättre och bättre anpassad för att korrekt skilja på de två klasserna.

2.3 Support Vector Machine

En annan enkel algoritm som kan användas för både regression och klassifikation är Support Vector Machine. Målet med SVM är att hitta det hyperplan som optimalt särskiljer datapunkter i rummet $\mathbb{R}^n$, där n är antal egenskaper för indata. Vid separation av datapunkter i två klasser finns det flera hyperplan som kan väljas. Med SVM väljs det plan som maximerar avståndet till de närmsta datapunkterna i båda klasser. Det hyperplan som bestäms kallas *Decision Boundary* (DB) och används sedan för att klassificera datapunkter genom att avgöra vilken sida om DB de hamnar på.

För att hantera datapunkter som inte kan separeras av en linjär DB används så kallade kärnfunktioner. Idén bygger på att först hitta en funktion ϕ som transformerar datapunkterna så att det sedan blir möjligt att separera dem med en linjär gräns. Problemen som uppstår är att ϕ ofta är okänd och att kostnadsfunktionen för SVM kräver en kostsam beräkning av den inre produkten $\phi(x_i) \cdot \phi(x_j)$ mellan två indataexempel i och j. Genom att definiera den inre produkten som en icke-linjär kärnfunktion $K(x_i, x_j)$ blir beräkningen mindre kostsam och det behöver inte tas reda på hur funktionen ϕ ser ut.

Som kärnfunktion väljs ofta den radiella basfunktionen (RBF):

$$K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2} \tag{2.4}$$

där γ är en valbar parameter. Vid användning av SVM förekommer även en valbar regulariseringsparameter C som kan varieras för att motverka överanpassning av modellen. Överanpassning innebär att modellen anpassas väldigt väl till träningsdata, men inte generaliserar väl för tidigare oanvänd data.

2.4 Artificiella neurala nätverk

Av den relativt enkla strukturen i logistisk regression följer att modellen inte presterar optimalt för klassifikation av mer komplexa mönster. Därför introduceras med fördel artificiella neurala nätverk. ANN bygger vidare på principerna som presenterades i avsnitt 2.2 om logistisk regression, men skiljer sig i användning av flera lager av vikter.

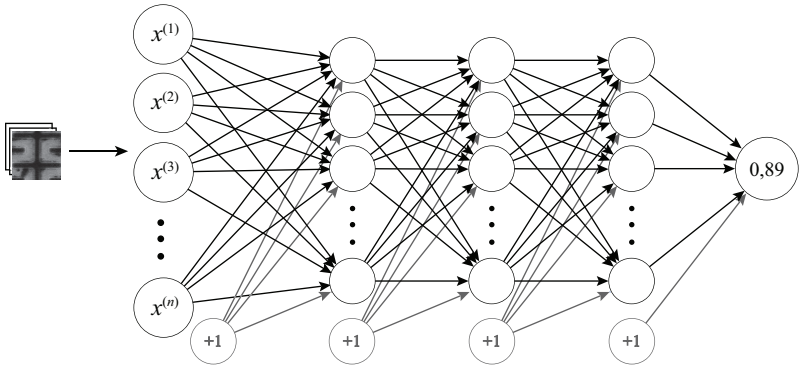
På senare år har ANN blivit allt mer populära och presterar på många sätt bättre än traditionella ML-algoritmer. Den ökade prestationen härstammar till stor del från den ökade komplexiteten av modellen [6]. Ett mer komplext nätverk kan lära sig mer komplexa beteenden, men kräver typiskt mer data, har fler hyperparametrar och kräver mer beräkningskapacitet. Vanliga neurala nätverk består av många enkla processorer, så kallade neuroner. Neuronerna är sammanbundna av viktade kopplingar som kallas kanter. I det enklaste fallet aktiveras neuroner av reella signaler och producerar sedan reella aktiveringssignaler genom en icke-linjär transformation, exempelvis den logistiska funktionen

(2.1). Ingångsneuroner aktiveras av reella indatavärden och övriga neuroner aktiveras genom tidigare aktiverade neuroner som är kopplade via en kant.

För att iterativt beräkna eller estimera gradienten för kostnadsfunktionen med avseende på nätverkets vikter, tillämpas ofta *bakåtpropagation*. Med bakåtpropagation beräknas gradienten lager för lager, genom att börja från sista lagret och gå bakåt. Algoritmen går bakåt eftersom framåtriktningens partiella derivator kan återanvändas för att beräkna bakåtriktningens. Med bakåtpropagation kan varje lager optimeras via Gradient Descent. Ofta används varianten Stochastic Gradient Descent, som vid varje iteration gör ett slumpmässigt urval av träningsdatan för att estimera gradienten. För teorin presenterad i detta stycke hänvisas till A. Ng, föreläsning 9.2-9.3 [5].

2.4.1 Dense Neural Networks

Dense Neural Networks (DNN) är en sorts neurala nätverk som består av neuronlager som är fullständigt sammankopplade. Att lagren är fullständigt sammankopplade innebär att alla neuroner i ett lager är direkt kopplade till alla neuroner i nästa lager, se Figur 2.2. I DNN tränas nästföljande lager på en kombination av alla egenskaper från tidigare lager.



Figur 2.2: Modellrepresentation av ett DNN med tre dolda lager. Bilder på mikrovågledarkretsar tolkas genom att pixelintensiteten plattas ned till en vektor och viktas med parametrar i flera lager. Som i Figur 2.1 flaggas kretsen som defekt, men den ökade komplexiteten i nätverkets struktur kan medföra att utdatalagret är än mer säkert på att kretssektionen är defekt. Därav följer den högre siffran 0,89 istället för 0,73. För varje lager tillkommer ytterligare en biasterm, som inte är sammankopplad med tidigare neuroner. Varje pil i figuren motsvarar en inlärd parameter.

Ett lager i DNN benämns ofta som just *Dense*-lager och är endimensionellt, som syns i Figur 2.2. För att Dense-lagret ska ta emot en bild behövs därför först ett lager som plattar ut bilden, ett så kallat *Flatten*-lager. Med exemplet 50×50 pixlar och gråskala åstadkommer Flatten-lagret att dimensionen ändras från $50 \times 50 \times 1$ till 1×2500.

2.4.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) är en typ av neurala nätverk som lämpar sig väl vid analys av bilder [7]. CNN har vanligen Dense-lager, men även ett eller flera lager som använder sig av faltning med mindre matriser än bildens dimensioner, så kallade filter. Ett faltningslager kallas *Conv*-lager. Filtren i Conv-lager är typiskt av dimension 3×3

eller 5×5. Faltningen innebär att ett filtret translateras över bilden, och inre produkter utförs med filtret och delmatriser av bilden. Värden som sammanfaller i bilden och filtret multipliceras alltså med varandra. Produkterna summeras, och varje position för filtret blir ett nytt matriselement. Resultatet blir en matris som kallas för egenskapskarta.

Filtren kan vanligtvis lära sig att detektera kanter, hörn, eller liknande relevanta egenskaper hos bilden. När exempelvis en kant stöts på av ett filter tränat för kantdetektion, ger det utslag i egenskapskartan genom att värdet sticker ut från resten av värdena i egenskapskartan. Exempelvis om värdet är betydligt högre jämfört med resten av egenskapskartan är det en indikation om att en kant har identifierats. För teorin i detta delavsnitt hänvisas till I. Goodfellow et al., kapitel 9 [8].

2.4.3 Transfer learning

Transfer Learning (TL) är en metod för att återanvända ANN som redan är tränade på en viss sorts data för att kunna lösa andra problem. Återanvändningen av det redan tränade nätverket görs genom att fixera en majoritet av parametrarna för att sedan träna om en mindre del, toppen av nätverket, så att det kan identifiera de nya egenskaperna i indatan. TL har fördelen att väldigt stora och vältränade ANN kan återanvändas utan att behöva träna om alla parametrarna, vilket kräver mycket beräkningskraft. ResNetV2 är ett exempel på ett neuralt nätverk som är tränat på den kategoriserade bilddatabasen ImageNet, och används ofta för TL-baserade metoder [9].

2.4.4 Pooling & Upsampling

Något som tillämpas i nästan alla CNN är ett lager som kallas *Pooling*-lager, vilket hjälper egenskapskartan till att bli nästintill invariant mot små translationer av indata. Med till exempel ett Maxpooling-lager translateras ett fönster över egenskapskartan och ersätter varje delmatris med det maximala pixelvärdet i delmatrisen. Med samma syfte finns även Global Average Pooling (GAP) där hela egenskapskartan ersätts med dess medelvärde.

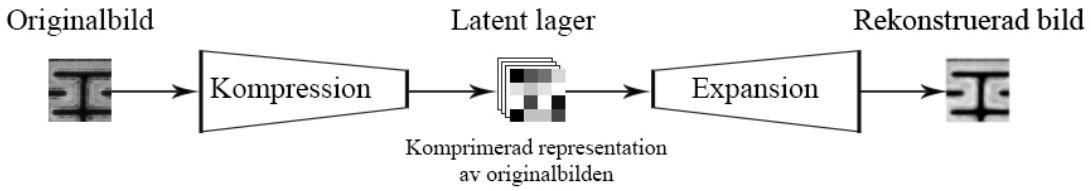
Motsatsen till Pooling är *Upsampling* som kan användas för öka den rumsliga dimensionen på egenskapskartan. Varje värde i egenskapskartan utökas då till en matris, vars storlek bestäms genom en valbar heltalsparameter. Teorin om Pooling och Upsamplinglager är hämtad från I. Goodfellow et al., kapitel 9 [8].

2.5 Anomaly Detection och Autoencoders

Om en av klasserna som ämnas klassificeras med hjälp av ett neuralt nätverk är kraftigt underrepresenterad, används med fördel Anomaly Detection (AD), som exempelvis tillämpas av K. Leung och C. Lecky i [10]. AD går ut på att detektera avvikelser i en datamängd. Ett sätt att tillämpa AD är att låta ett neuralt nätverk tränas på enbart den överrepresenterade klassen, så kallat *Novel Detection*. I fallet med mikrovågledarkretsar är rimligen klassen med ickedefekta kretsar överrepresenterad. Nätverket kommer vid träning på ickedefekta kretssektioner att vara väl anpassat för att tolka sådana kretsar, men om nätverket ges en bild på en defekt krets kommer sannolikt felet som ges av kostnadsfunktionen vara högt. Ett AD-nätverk kan även tränas på all data, inklusive den defekta

[10]. Principen vilar då just på att en klass är underrepresenterad och under träning inte tillför mycket information att dra slutsatser från, såsom förutsättningarna är i [10]. Träning sker då helt utan vetskap om klasserna som bilderna tillhör genom att ta bort annoteringarna. Maskininlärning utan annoteringar kallas för *Unsupervised Learning*.

En särskild sorts neuralt nätverk som kan implementeras för AD är *Autoencoder* (AE). Ett AE-nätverk tar in en bild på en kretssektion och komprimerar den till ett mindre format med hjälp av flera lager med neuroner. Efter kompressionen expanderas den komprimerade representationen till att efterlikna ursprungsbilden [11]. Rekonstruktionen sker med ytterligare lager av neuroner. Se I. Goodfellow et al., kapitel 14 för teori om AE, där den komprimerande respektive expanderande delen kallas för *encoder* respektive *decoder* [8]. I Figur 2.3 illustreras strukturen för ett AE-nätverk.



Figur 2.3: Illustration av AE med en ickedefekt kretssektion som indata. Bilden komprimeras till ett mindre format, vilket ofta kallas det latenta lagret. Om nätverket tränats på övervägande ickedefekta kretssektioner kan en väl rekonstruerad bild erhållas. Den "rekonstruerade" bilden är här originalbilden som har behandlats med en bildredigerare för att simulera en rekonstruktion.

Ett mått på hur väl AE:n bygger upp kretsen från den komprimerade formen är rekonstruktionsfelet [11]. Rekonstruktionsfelet avser hur väl den rekonstruerade bilden stämmer överens med den ursprungliga bilden. En ickedefekt kretssektion ska med AE kunna rekonstrueras med lågt rekonstruktionsfel om indata övervägande innehållit ickedefekta kretssektioner. Däremot avviker en defekt kretssektion från vad AE:n till större del tränats på att rekonstruera. Därför ger de avvikande defekta kretssektionerna upphov till ett högt rekonstruktionsfel. Det höga rekonstruktionsfelet kan då användas för att separera defekta kretssektioner från ickedefekta. På liknande sätt som andra neurala nätverk kan lagren för en Autoencoder se olika ut. På samma sätt som CNN lämpar sig Autoencoders med Conv-lager väl för bildanalys. En AE med Conv-lager kallas Convolutional Autoencoder (CAE).

2.6 Aktiveringsfunktioner och kostnadsfunktioner

Hittills har i avsnitt 2.2 aktiveringsfunktionen och dess tillhörande kostnadsfunktion för logistisk regression berörts, men det finns fler exempel som flitigt används inom ML. Det kanske vanligaste exemplet på en aktiveringsfunktion inom djup maskininlärning är den styckvis linjära ReLU (Rectified Linear Unit), som har tillämpats av exempelvis Alex Krizhevsky *et al.* [12]. ReLU påstås vara en viktig milstolpe inom djup maskininlärning [13]. Funktionen används oftast i modellens dolda lager, inte som aktiveringsfunktion för sista lagret [13]. I binär klassifikation är det lämpligt att använda sigmoid (2.1) i sista lagret, eftersom funktionen ger ett värde mellan 0 och 1. Aktiveringsfunktionen $g(\mathbf{z}^l) =$

$\text{ReLU}(\mathbf{z}^l)$, med $\mathbf{z}^l$ som utdata för lager l, definieras som

$$\text{ReLU}(\mathbf{z}^l) = \max(0, \mathbf{z}^l). \tag{2.5}$$

Eftersom $\text{ReLU}(\mathbf{z}^l) = 0$ för $\mathbf{z}^l \leq 0$ innebär det att många neuroner i ett ANN är inaktiva och alltså ger värdet 0 för given indata. Det innebär i sin tur att neuronernas utdata kan representeras av en matematiskt gles matris. Glesa matriser medför flera beräkningsmässiga fördelar, vilka utförligt diskuteras av X. Glorot *et al.* [14]. Vidare är ReLU en mindre beräkningstung funktion än sigmoid (2.1), som kräver kostsamma exponentiella beräkningar [14]. De beräkningsmässiga fördelarna med ReLU möjliggör djupare maskininlärning, och aktiveringsfunktionen har observerats ge mycket god prestation inom ML-modeller för mönsterigenkänning [13].

Även vid djupare nätverksarkitekturer som tillämpar binär klassifikation förekommer användning av kostnadsfunktionen för logistisk regression (2.3), eller någon variant av den [15]. I binär klassifikation kallas den för BCE (Binary Cross-Entropy). En annan vanlig kostnadsfunktion är MSE (Mean Squared Error), som definieras via

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{m} \sum_{i}^{m} (y_i - \hat{y}_i)^2, \tag{2.6}$$

där $\mathbf{y}$ är träningsdatas värde, $\hat{\mathbf{y}}$ är nätverkets förutspåelse på indata $\mathbf{x}$ och m är antal element i $\mathbf{y}, \hat{\mathbf{y}}$. För exemplet med en AE kan MSE tillämpas för att kvantifiera felet mellan rekonstruktionerna och originalbilderna, där m är pixlarna hos bilderna [11]. MSE ligger mellan 0 och 1 för pixelintensiteter angivna som flyttal mellan 0 och 1.

2.7 Datauppsplittring

Detta delavsnitt utgår från C. M. Bishop [4] samt A. Ng [5] (föreläsning 10.3). Inom maskininlärning är det praxis att dela upp data i en viss andel tränings-, valideringss- och testdata. *Träningsdatan* är den data som används för att anpassa modellens parametrar genom minimering av kostnadsfunktionen. Större delen av datan är vanligen träningsdata. Antalet fullständiga iterationer av träning kallas för *epoker*.

För att under tiden av träning evaluera hur väl modellen generaliserar mot en datauppsättning som är opartisk mot träningsdatan, används *valideringsdatan*. Valideringsdata kan exempelvis användas för att bedöma vid vilken punkt träning bör avbrytas vid minimering av kostnadsfunktionen, eller för att avgöra vilken av flera konkurrerande modeller som bör väljas. Typiskt önskas både kostnaden för träningsdatan och valideringsdatan minimeras. Minimering av kostnad för valideringsdata sker med rätt implementation som biprodukt av träning på träningsdatan. Modellen tränas på träningsdatan, medan valideringsdatan är ett medel för användaren att finslipa hyperparametrarna.

Slutligen används *testdatan*, som är en sista bedömning av hur väl modellen presterar efter en fullständig träning. Testdatan bekräftar generaliseringen av modellen efter träning med träning- och valideringsdata. Exempelvis används testdatan vid binär klassifikation vanligen för att se hur stor andel datapunkter av respektive sort klassificeras korrekt. En vanlig uppdelning är 60 % träningsdata, 20 % valideringsdata och 20 % testdata.

3 Metod

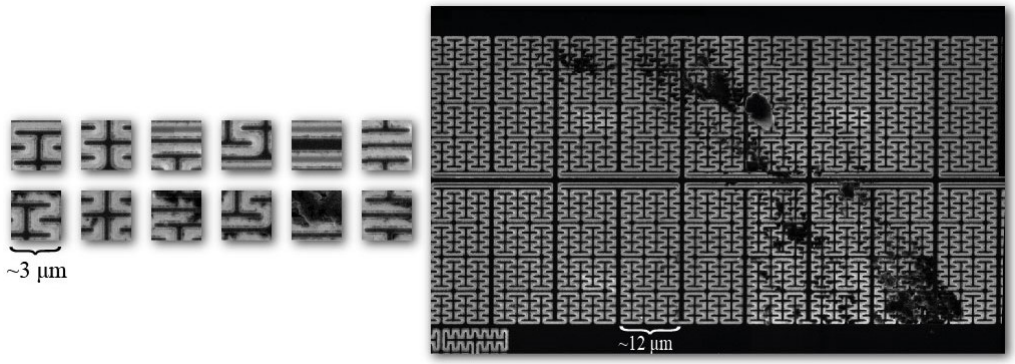
I detta avsnitt presenteras vilka programvaror som användes och konstruktion av nätverksarkitekturen för de tillämpade maskininlärningsmodellerna. Vidare introduceras de olika datamängder som tränades på under studien, och vilka ML-modeller som tillämpades för vardera datamängd.

3.1 Programvaror för maskininlärning och bildbehandling

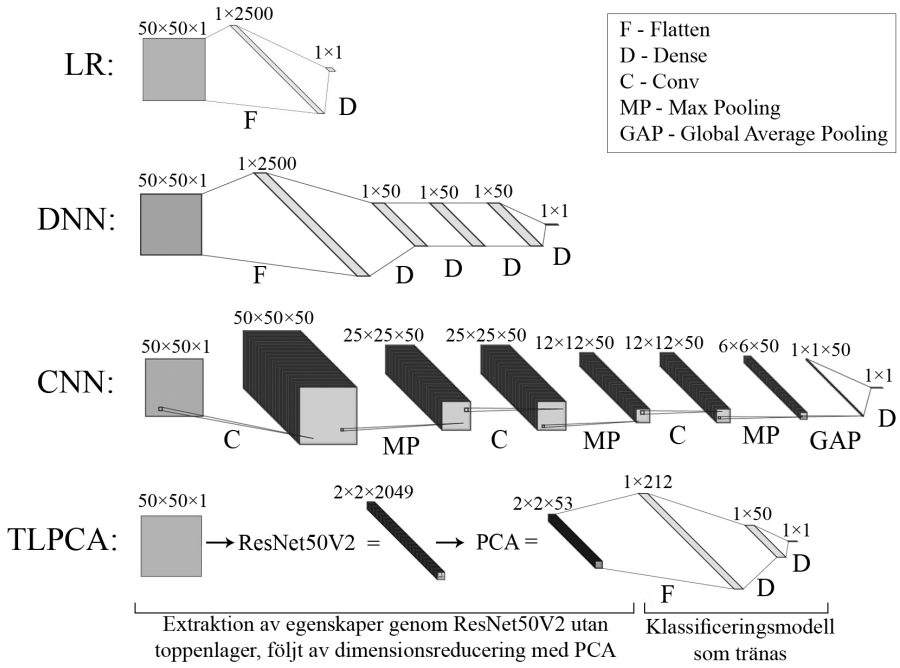
För att konstruera och träna nätverken användes programmeringsspråket Python, huvudsakligen med TensorFlow [16] och Keras [17]. Vidare användes Scikit-learn [18] för diverse modellbehandling såsom uppsplittring av dataseten i tränings-, validerings- och testdata. Programvarubiblioteket OpenCV [19] brukades vid flera tillfällen för att läsa in bilder, skapa och augmentera datamängder. För att koordinera arbetet för flera medverkande parter och ha tillgång till versionshistorik användes GitHub [20], som är en digital värd för mjukvaruutveckling. För att spara resultatet vid träning av nätverken i avsnitt 3.4.1 användes Weights & Biases [21], som är ett verktyg för att organisera och analysera ML-tester.

3.2 Balanserad datamängd

Den första datamängden är ett exempel på en tidig generation av mikrovågledarkretsar. Datamängden benämns Gen1 (första generationen). Gen1 bestod av högkontrastbilder på sektioner om 50×50 pixlar av en mikrovågledarkrets med en mycket jämn fördelning av 1251 defekta respektive 1235 ickedefekta kretssektioner i formatet TIFF. Formatet användes främst eftersom det är ett av möjliga okomprimerade filformat för bilder tagna med SEM i renrummet på MC2. Ett urval av bilder på mikrovågledarkretsen syns i Figur 3.1. Kretsen innehöll ovanligt många defekter jämfört med andra kretsar och var därmed inte särskilt representativ för vanliga fall av mikrovågledarkretsar. Dessa typer av defekter, där stora församlingar av orenheter uppstår, är vanligen ett användarfel. Huvudsyftet med datamängden var att testa och utveckla förståelse för olika ML-modeller, för att evaluera möjligheterna för senare datamängder.



Figur 3.1: **(a)** är ett urval av sektioner om 50×50 pixlar på mikrovågledarkretsen Gen1. Den övre raden är bilder på ickedefekta delar av kretsen, medan den undre raden är bilder på defekta delar. **(b)** är en större del av kretsen som är särskilt defekt.



Figur 3.2: Modellrepresentation för samtliga modeller som användes för Gen1, skilt från SVM, som inte är nätverksbaserad. Alla dolda lager i CNN-modellen samt DNN-modellen och det näst sista lagret i TLPCA har aktiveringsfunktionen ReLU. Alla utdatalager har aktiveringsfunktionen sigmoid. Bilden genererades med webverktyget "NN-SVG" av Alex LeNail [22].

3.2.1 Tillämpade modeller

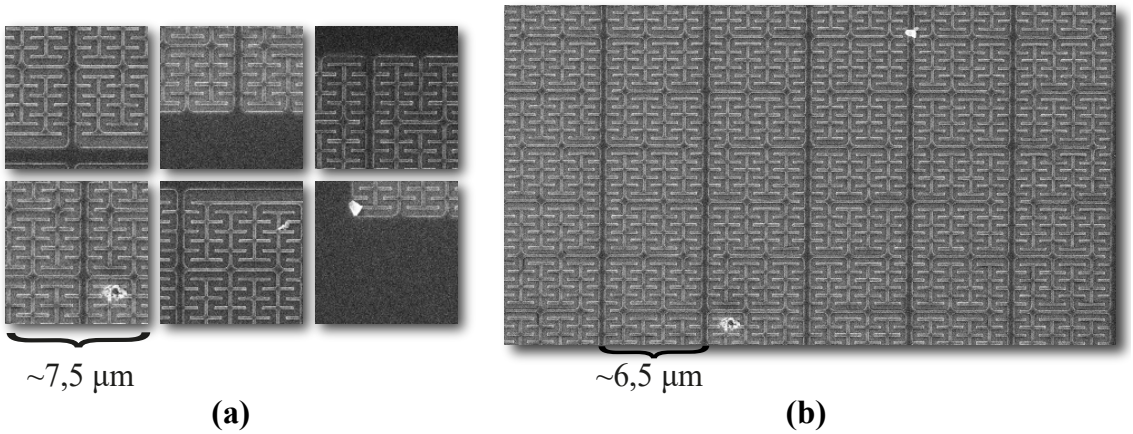
För det balanserade datasetet tillämpades huvudsakligen ett antal nätverksbaserade modeller. De modeller som var nätverksbaserade är illustrerade i Figur 3.2. Första modellen (LR) står för logistisk regresssion med hjälp av ett Dense-lager. Utöver de nätverksbaserade modellerna användes även en SVM-baserad modell med RBF som kärnfunktion, se ekvation (2.4). Parametrarna valdes till $\gamma = 0.001$ och $C = 100$. I den TL-baserade modellen TLPCA reduceras antalet träningsbara parametrar med hjälp av *Principal Component Analysis* (PCA). PCA används för att reducera dimension på indata, samtidigt som så mycket information som möjligt är bevarad. På så sätt blir modellen mindre komplex och träning av modellen kan optimeras [8].

För att genomföra bakåtpropagation med modellerna LR, CNN och TLPCA användes optimeringsmetoden Adam [23]. Som kostnadsfunktion användes BCE, se ekvation (2.3). Modellernas prestation kvantifierades genom att betrakta varje modells noggrannhet, som är kvoten mellan korrekt förutspådda bilder och totala antalet bilder i procentenheter.

3.3 Obalanserad datamängd med underrepresenterade defekter

Efter behandling av den balanserade datan undersöktes en ny datamängd (Gen2) med obalanserad fördelning av bilder på defekta respektive ickedefekta kretssektioner, fångade med SEM och av format TIFF av samma motivering som tidigare. Gen2 bestod av bilder med lägre kontrast, på sektioner om 128×128 pixlar av en mikrovågledarkrets med extremt obalanserad fördelning av 22 defekta respektive 9126 ickedefekta kretssektioner. Ett urval av bilder på Gen2 illustreras i Figur 3.3, som visar att defekterna först och främst

förekommer i form av orenheter som syns som vita fläckar på kretsarna. Gen2 är ett realistiskt fall av fördelningen och utseendet av tillverkningsdefekterna som förekommer vid tillverkning av mikrovågledarkretsarna på MC2.



Figur 3.3: **(a)** visar sektioner om 128×128 pixlar från Gen2. Den övre raden är bilder på ickedefekta delar av kretsen, medan den undre raden är bilder på defekta delar. **(b)** visar en större del av kretsen som är särskilt defekt. Notera att tätheten av defekter på Gen2 är betydligt lägre än för Gen1.

Eftersom fördelningen av defekter och ickedefekta sektioner i Gen2 var extremt obalanserad misstänktes att modellerna i föregående avsnitt för Gen1 skulle få svårt att lära sig skillnaden mellan defekt och ickedefekt bild. För att utreda misstanken gjordes några försök att träna modellerna, Figur 3.2 & SVM, likt som på Gen1. Det noterades först att bilderna skiljde sig i storlek jämfört med Gen1. För att enklare kunna jämföra tillämpningarna på Gen1 och Gen2 gjordes en förprocessering genom att beskära bilderna till sektioner om 64×64 pixlar. Bilder på ickedefekta kretssektionerna fyrdubblades då till 36505, men antal defekta kretssektioner var fortfarande 22. Vid datauppsplittringen säkerställdes sedan att de 22 defekta kretssektionerna fördelades ut enligt proportionerna för tränings-, validerings- och testdata som nämndes i teoriavsnittet 2.7.

Försöken med modellerna LR, DNN, CNN, TLPCA (Figur 3.2) och SVM resulterade endast ickedefekta kretssektioner förutspåddes vid klassificering av testdata, vilket gjorde det tydligt att modellerna endast lärde sig fördelningen i Gen2. Därför avfärdades de första modellerna och andra metoder för obalanserad data började undersökas. För att överkomma svårigheten med obalanserad data föreslogs två tillvägagångssätt:

1. **Dataaugmentering:** Utöka defekta kretssektioner genom transformationer på befintlig data.

2. **Convolutional Autoencoder:** Träna en CAE på ickedefekta kretssektioner och förutspå defekt eller ickedefekt med hjälp av ett tröskelvärde för rekonstruktionsfelet.

3.3.1 Dataaugmentering

Att utöka den underrepresenterade klassen är en möjlig metod för att hantera obalanserad data. Utökningen ska endast utföras på träningsdata för den underrepresenterade klassen,

eftersom modellen annars riskerar att memorera bilder på defekta kretssektioner istället för att lära sig egenskaper för defekterna.

För att testa metoden fokuserades det främst på geometriska transformationer som translation, spegling och rotation. På grund av symmetrin hos fraktalerna är de nämnda transformationerna rimliga då de producerar bilder som kan tänkas synas i Gen2. Här gjordes också separat fördelning av de 22 defekta bilderna vid datauppsplittringen, vilket ger 14 bilder till träningsdata att utöka.

Tabell 3.1: Utökning av defekta kretssektioner i träningsdata genom geometriska transformationer. Varje kolumn motsvarar hur många bilder som genererats efter varje transformation.

Initialt antal	Translation	Vertikal spegling	Horisontell spegling	Rotation $180°$
14	140	280	560	1120

I Tabell 3.1 visas utökningen av defekta kretssektion i träningsdata. Först genereras 10 slumpvisa translationer kring defekten för varje varje bild. Därefter transformeras bilderna genom vertikal och horisontell spegling samt $180°$-rotation, vilket adderas till redan utförd transformation, därför dubbleras antalet bilder i varje steg. Antalet ickedefekta kretssektioner valdes sedan till att motsvara antalet defekta kretssektioner i träningsdata. Den totala fördelningen visas i Tabell 3.2.

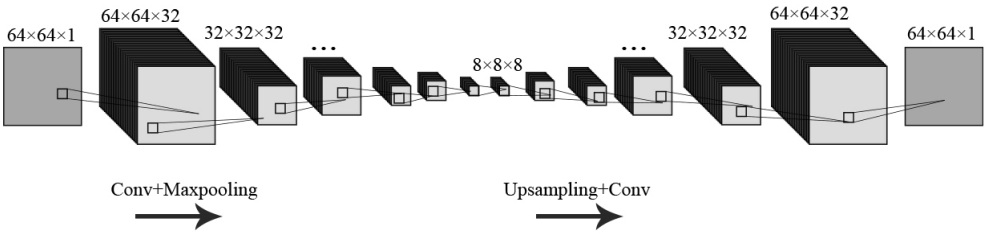
Tabell 3.2: Fördelning av defekta och ickedefekta kretssektioner i tränings-, valideringss- och testdata med utökade defekta kretssektioner.

Kategori	Träning	Validering	Test
Defekta	1120	4	4
Ickedefekta	1120	373	373

Försök gjordes genom att träna modellen CNN, Figur 3.2, på Gen2. Resultatet vid klassifikation av testdata var likt tidigare att alla bilder klassificerades som ickedefekta. Några försök gjordes även genom att dessutom utöka antalet defekta kretssektioner i valideringsdata genom 10 slumpvisa translationer. Även det försöket resulterade i samma utfall som tidigare och det ansågs därför att obalansen i Gen2 förmodligen var för extrem för att kunna lösas med augmentering.

3.4 Tillämpning av CAE på obalanserad datamängd

För att tillämpa CAE på Gen2 testades först en modell med nätverksarkitektur som illustreras i Figur 3.4. Arkitekturen är inspirerad av F. Chollet [24].



Figur 3.4: Nätverksarkitektur för det första tillämpade CAE-nätverket. Nätverket komprimerar en bild genom en serie av $3\times$(Conv+Maxpooling-lager). En rekonstruerad bild genereras sedan genom en serie av $3\times$(Conv+Upsampling-lager), samt ett avslutande Conv-lager, som expanderar den komprimerade representationen av bilden. Samtliga Conv-lager har aktiveringsfunktion ReLU, förutom det sista lagret. Sigmoid används för det sista lagret för att ge ett pixelvärde mellan 0 och 1. Notera att indata är fortsatt de beskurna bilderna om 64×64 pixlar från Gen2.

Vid denna del av studien blev träning av nätverken på egna datorer mycket krävande för processenheterna. För att accelerera träningen och möjliggöra en bredare bas av testade nätverksarkitekturer användes PC-klustren Vera och Hebbe på Chalmers Centre for Computational Science and Engineering (C3SE), som är ett av de sex nationella metacenter som tillhandahålls av Swedish National Infrastructure for Computing (SNIC).

3.4.1 CAE – Novel Detection

Som beskrivet i 2.5 kan CAE användas för att hitta avvikelser i datamängder genom att endast träna ett nätverk på den överrepresenterade klassen, för att sedan särskilja mot den underrepresenterade klassen med hjälp av ett rekonstruktionsfel. Med Gen2 valdes därför att träna den framtagna CAE-modellen (Figur 3.4) i 100 epoker på 80 % av de ickedefekta sektionerna på 64×64 pixlar. De resterande 20 % användes till testdata tillsammans med de defekta kretssektionerna. Här gjordes en utökning av de defekta sektionerna genom 3 translationer per bild, vilket alltså resulterade i 66 defekta sektioner. Utökningen gjordes för att se hur rekonstruktionsfelet ändrades beroende på var en defekt var placerad i bilden.

Ett sätt att mäta nätverkets rekonstruktionsfel på testdata togs fram genom att beräkna MSE över mindre sektioner i bilden. Principen förklaras utförligare i 3.4.3. Resultatet (se avsnitt **??**) visade att en majoritet av de defekta och ickedefekta sektionerna var tydligt särskiljbara i deras rekonstruktionsfel.

Efter att ha testat att ändra några av nätverkets hyperparametrar konstaterades att rekonstruktionen av bilderna var väldigt beroende av dimensionen för det latenta lagret, se Figur 2.3. Ett försök att optimera nätverket avgränsades därför till att variera antal filter för Conv-lagren, samt att testa ett förlängt nätverk av det i Figur 3.4. Det förlängda nätverket hade strukturen av $4\times$(Conv+Maxpooling-lager) med efterföljande $4\times$(Conv+Upsampling-lager), samt ett avslutande Conv-lager. I Tabell 3.3 visas antal filter som testades för det initiala nätverket respektive det förlängda.

Tabell 3.3: Testade antal filter per lager med nätverket från Figur 3.4 i **(a)** och den förlängda varianten i **(b)**. De testade nätverken med olika antal filter per lager är angivna som Modell 1-8. Notera att Modell 2 är det nätverk som är illustrerat i Figur 3.4.

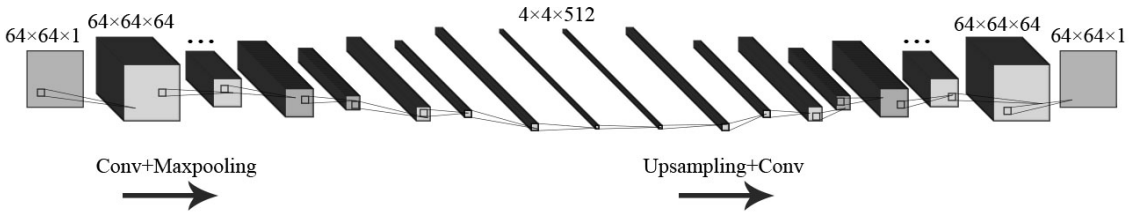
Modell	Lager 1-3	Lager 4-6
1	64, 32, 16	16, 32, 64
2	32, 16, 8	8, 16, 32
3	16, 8, 4	4, 8, 16
4	64, 128, 256	256, 128, 64

(a) Antal filter per lager som testades med nätverket enligt Figur 3.4.

Modell	Lager 1-4	Lager 5-8
5	64, 32, 16, 8	8, 16, 32, 64
6	32, 16, 8, 4	4, 8, 16, 32
7	16, 8, 4, 2	2, 4, 8, 16
8	64, 128, 256, 512	512, 256, 128, 64

(b) Antal filter per lager som testades med det förlängda nätverket.

De flesta modellerna som testades följde samma princip som den första modellen, Figur 3.4, där både rumsdimension och antal filter är komprimerat för det latenta lagret. Utöver det testades även att låta antalet filter öka in mot mitten av nätverket, vilken är redovisat som Modell 4 och Modell 8 i Tabell 3.3. Träningen gjordes likt innan under 100 epoker. Modell 8 visade sig vara den modell som bäst separerade rekonstruktionsfelet för defekta och ickedefekta kretssektioner, se resultat i avsnitt 4.2.1. Modell 8 är illustrerad i Figur 3.5.



Figur 3.5: Nätverksarkitektur för Modell 8. Likt förra modellen har samtliga Conv-lager aktiveringsfunktion ReLU, förutom den sista, som använder sigmoid.

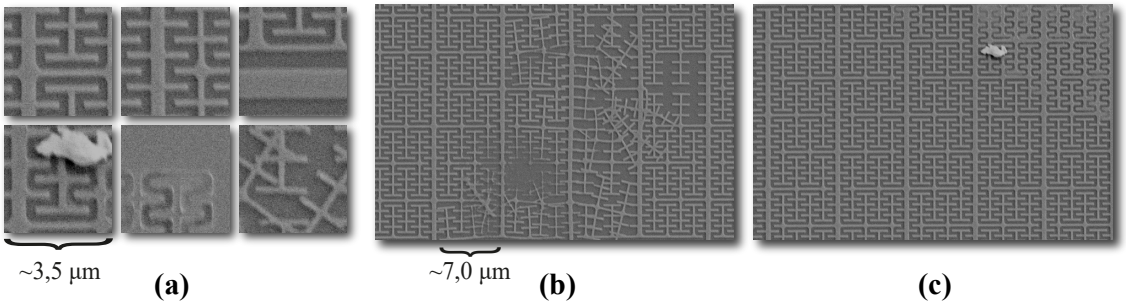
3.4.2 CAE – Unsupervised Learning

I teoriavsnittet 2.5 nämns även möjligheten att tillämpa CAE på helt oannoterad datamängd för att upptäcka avvikelser. Om tillvägagångssättet är möjligt skulle det innebära att mindre arbete behöver utföras för att annotera bilder i förhand. Därför testades i nästa steg att träna det optimerade CAE-nätverket Modell 8, Figur 3.5, på alla bilder som samlats in på kretsen som förekom i Gen2 under 100 epoker. Koden för modellen finns i Appendix B. Bilderna var i råformat 3072×2304 pixlar och plockades ut i kretssektioner om 64×64 pixlar. Ett överlapp på 16 pixlar över kretssektionerna valdes då det visade sig att defekter som låg precis på kanten av en kretssektion tenderade ge lägre rekonstruktionsfel. Uppdelning resulterade i en datamängd med 104448 bilder som benämndes Gen2*. Mer information om Gen2* finns i Tabell A.1.

För att kunna träna på ytterligare fall av förekommande bilder på mikrovågledarkretsar fångades nya TIFF-bilder med SEM i renrummet på MC2. Samlingen bilder benämns Gen3. Bilderna var som innan i råformat storleken 3072×2304 pixlar. Uppdelning gjordes till sektioner om 128×128 pixlar för att inkludera liknande mycket fysiskt innehåll

som Gen2* och hade överlapp om 32 pixlar. Bilderna samplades sedan ned till 64×64 pixlar för att kunna använda samma modellstruktur som för Gen2*. Uppsättningen bilderna bestod likt Gen2 av ytterst få defekta kretssektioner jämfört med ickedefekta. Det syntes även nya sorters defekter på kretssektionerna, som illustreras i Figur 3.6. Träning av Modell 8 på Gen3 gjordes likt innan i 100 epoker.

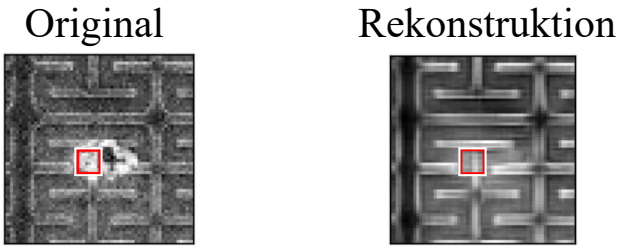
Till skillnad från Gen1 och Gen2, där alla bilderna var annoterade som defekta eller ickedefekta, saknades alltså annotering helt för Gen2* och Gen3 då dessa datamängder skulle användas för Unsupervised Learning av CAE. Datauppsplittringen som användes var nu istället 70 % träningsdata och 30 % valideringsdata. Efter träning av modellen tilläts alla bilder rekonstrueras av nätverket. Samtliga kretssektioner sorterades efter högst MMSE8 för att sedan undersökas manuellt.



Figur 3.6: **(a)** visar sektioner om 128×128 pixlar på Gen3. Den övre raden är ickedefekta sektioner, den undre raden defekta. **(b)** och **(c)** visar större områden med olika sorters defekter. De olika sorters defekterna visas mer tydligt i **(a)**. Se även Tabell A.1 för en mer detaljerad beskrivning.

3.4.3 Prestationsmetrik

För varje rekonstruerad bild sveptes en ruta om $p \times p$ pixlar över bilden, där p är ett positivt heltal. En intermediär matris bildades med MSE för alla på $p \times p$ pixlar. Det maximala elementet i den intermediära matrisen identifierades, och valdes som metrik. Metriken benämndes MMSEp (Maximum Mean Squared Error, dimension p). Se Figur 3.7 för en illustration av metriken där $p = 8$. Positionen för maxfelet sparades och användes för att markera en förutsägelse om var defekten befann sig för defekta kretssektioner. Kretssektioner plottades i fallande ordning av MMSEp. För Gen2* och Gen3 stegades rutan med 4 pixlars mellanrum för att minska beräkningstiden, jämfört med 1 pixels steglängd för Gen1 och Gen2. Steglängden observerades ha låg påverkan på resultatet.



Figur 3.7: En kretssektion som jämförs med dess rekonstruktion av en tidig modell av ett CAE-nätverk. Den rödvita rutan visar det 8×8-område vid vilket rekonstruktionsfelet är störst (MMSE8). Som förväntat utritas rutan där defekten ligger. Notera att CAE-nätverket har fyllt i det defekta området med vad som ser ut som oskadade kretssegment.

Metriken valdes huvudsakligen för att överkomma ett problem som upptäcktes med att använda MSE på testdatan. Bilder där stora områden var utanför kretsen bestod mestadels av brus av relativt jämn intensitet. Områden med brus rekonstruerades enkelt av CAE:n. Det låga rekonstruktionsfelet gav upphov till att en kretssektions MSE drogs ned kraftigt, även om det befann sig en defekt i bilden. Som diskuterades i avsnitt 2.5 skiljs bilderna åt genom deras rekonstruktionsfel. En bild på en defekt kretssektion med mycket stor del utanför kretsen riskerar därför att flaggas som ickedefekt med MSE.

MMSE*p* överkom problemet i alla observerade fall. För defekta sektioner var största rekonstruktionsfelet mitt på defekten, eller vid en del av defekten. Defekterna markerades därför tydligt av den utritade rutan, som är hjälpsamt för att snabbt visuellt bekräfta var defekterna befinner sig. Det oberverades en mycket tydligare skillnad av defekta kontra ickedefekta kretssektioner jämfört med att använda MSE. Eftersom nätverket är mycket bättre på att rekonstruera ickedefekta kretssektioner resulterar det i att det maximala felet förekommer vid mycket mer oförutsägbara områden hos ickedefekta kretssektioner. MMSE8 valdes för de slutgiltiga versionerna av CAE, då den observerades ge bäst resultat generellt, utav några olika testade värden för p.

4 Resultat

I detta avsnitt presenteras resultaten för studien i form av prestation av nätverken. I första respektive det andra delavsnittet presenteras prestationen för de olika ML-modellerna applicerade på Gen1 respektive Gen2, Gen2* och Gen3. Datamängderna finns sammanfattade i Tabell A.1. Resultaten analyseras kort, och utvecklas i diskussionsavsnittet.

4.1 Modellernas prestation för balanserad data

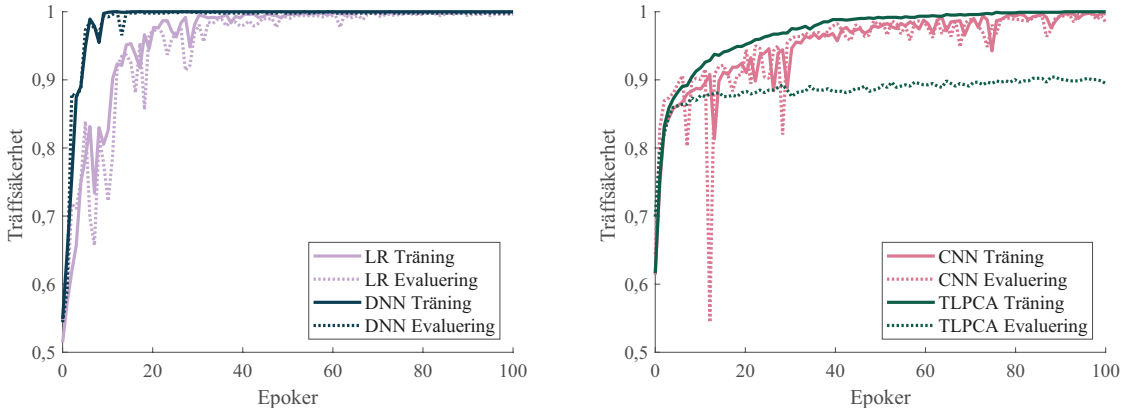
På Gen1 lyckades alla modeller förutom TLPCA och SVM nå upp till 1,0 i träffsäkerhet efter fullständig träning, vilket alltså innebär att alla bilder i testdatan klassificerades korrekt. LR och SVM behövde tränas under fler epoker än resterande modeller för att nå lika högt som DNN och CNN, men nådde 1,0 i träffsäkerhet efter 200 epoker. Delresultatet att LR nådde upp till 1,0 eller nära 1,0 i träffsäkerhet kan vid första anblick tyckas förvånande med tanke på simpliciteten i modellen. Det beror emellertid sannolikt på hög kontrast i bilderna, väldefinierade defekter och en balanserad fördelning av de två klasserna. TLPCA presterade samtidigt sämst, vilket kan tyda på överanpassning. Nätverksarkitekturen kan vara för avancerad för att fungera väl för denna datamängd. CNN-modellen presterade i snitt något sämre, möjligen av samma skäl som TLPCA men kan även ha krävt träning i fler epoker för att nå upp till samma prestation som DNN.

Skillnader syntes i reproducerbarheten i resultaten. Efter vissa fullständiga träningar felklassificerade modellerna några fåtal bilder och fick lägre träffsäkerhet. DNN nådde upp till träffsäkerhet 1,0 oftast och uppvisade överlag minst varians, vilket föreslår att DNN är att föredra framför de andra modellerna för Gen1. På grund av arten av problemet är det viktigt att defekter inte missas, då en enstaka defekt som orsakar kortslutning helt hindrar kretsen från att fungera. Resultat för samtliga modeller tränade under 100 epoker finns sammanställt i Tabell 4.1. Hyperparametrarna hos modellerna optimerades inte till

stor utsträckning, eftersom modellerna redan nådde upp till hög träffsäkerhet utan mycket optimering. Mestadels optimerades antal dolda lager och antal neuroner i DNN-modellen.

Tabell 4.1: Träffsäkerhet hos de olika modellerna samt andelen felklassificeringar. Värden för träffsäkerhet betecknas: genomsnitt $\bar{x}$, standardavvikelse σ, maxvärde x_{max}, och minimumvärde x_{min}. "Felklassificering (ickedefekt)" innebär att modellen har klassificerat en defekt kretssektion som ickedefekt och "Felklassificering (defekt)" innebär att modellen har klassificerat en ickedefekt sektion som defekt. Tio olika träningar på 100 epoker utfördes för respektive modell.

Modell	SVM	LR	DNN	CNN	TLPCA
$\bar{x}$	0,9952	0,9957	0,9993	0,9961	0,9201
σ	0,0022	0,0031	0,0007	0,0047	0,0099
x_{max}	0,9987	0,9987	1,0000	1,0000	0,9397
x_{min}	0,9920	0,9879	0,9987	0,9839	0,9062
Felklassificering (ickedefekt)	0,0000	0,0000	0,0000	0,0054	0,0311
Felklassificering (defekt)	0,0048	0,0043	0,0007	0,0034	0,0488



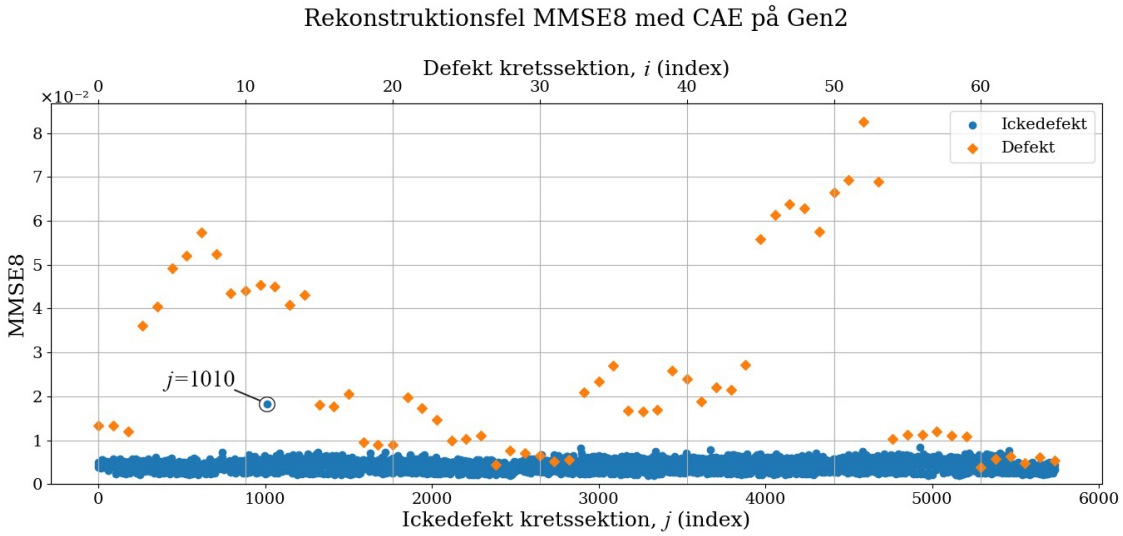
Figur 4.1: Exempel på träffsäkerhet per epok hos träning- och valideringsdata för en träning vardera av de olika nätverksmodellerna. Heldragna linjer representerar träffsäkerheten på träningsdatan och streckade linjer träffsäkerheten på valideringsdatan som valdes till 30 % av all data.

4.2 Prestation av CAE för obalanserad data

Nedan följer resultat som erhållits vid tillämpning av CAE genom Novel Detection samt Unsupervised Learning på datamängderna Gen2, Gen2* och Gen3. Varje delresultat evalueras till största del genom hur väl rekonstruktionsfelet MMSE8 separerar defekta och ickedefekta kretssektioner. Möjligheten att ta fram ett tröskelvärde för MMSE8 vid klassifikation av nytillkomna kretssektioner tas upp i diskussionsavsnittet 5.1.1.

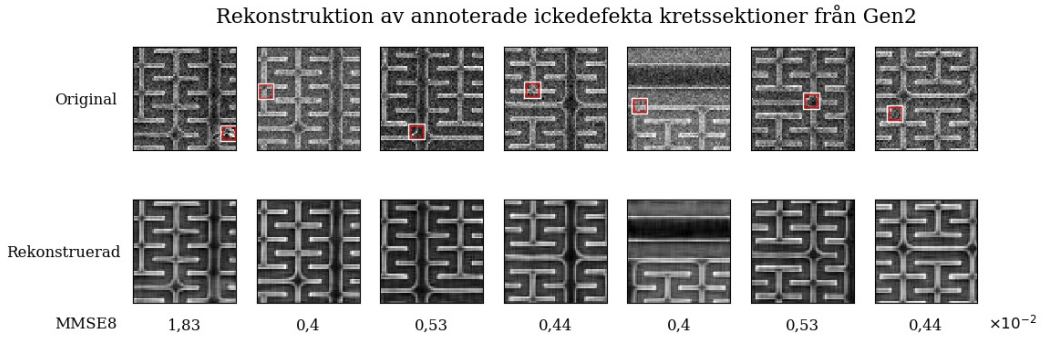
4.2.1 CAE – Novel Detection

I Figur 4.2 visas rekonstruktionsfelet MMSE8 för testdata i Gen2 med den initala CAE-modellen, Figur 3.4. En majoritet av testbilderna för de ickedefekta kretssektionerna har MMSE8 på ungefär $0{,}4 \cdot 10^{-2}$, vilket till stor del skiljer sig från de defekta kretssektioner som har MMSE8 från ungefär $0{,}4 \cdot 10^{-2}$ upp till $9 \cdot 10^{-2}$. Rekonstruktionsfelet MMSE8 lyckas därmed nästan totalt separera defekta kretssektioner från ickedefekta.



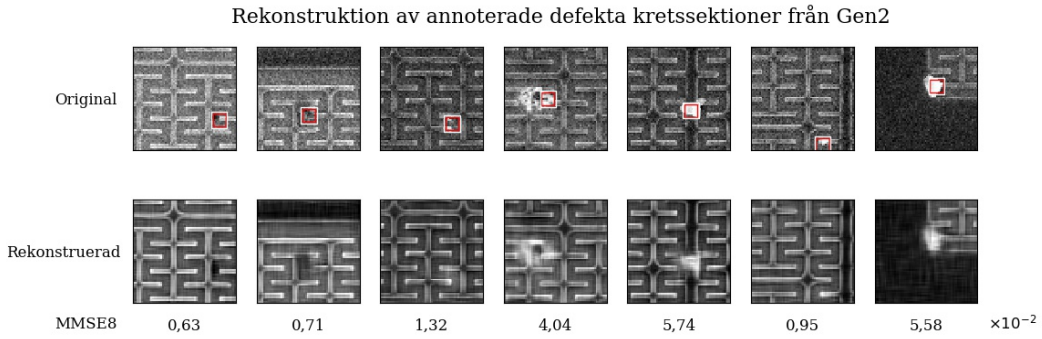
Figur 4.2: MMSE8 för testdata från Gen2 med det initiala CAE-nätverket, Figur 3.4. Index för defekt och ickedefekt testdata visas på varsin x-axel som i respektive j. Området med hög täthet av ickedefekt testdata med lågt rekonstruktionsfel kan betraktas som ett sorts brusgolv.

I Figur 4.2 finns några specifika noterbara datapunkter. En annoterad ickedefekt testbild, $j = 1010$, registrerades med ett betydligt högre rekonstruktionsfel än resterande ickedefekta testbilder. Vid granskning av bilden noterades att kretssektionen innehöll en defekt, som hade missats vid annoteringen av Gen2. Kretssektionen syns som första exemplet i Figur 4.3, där ett urval av rekonstruerade kretssektioner som annoterats ickedefekta visas.



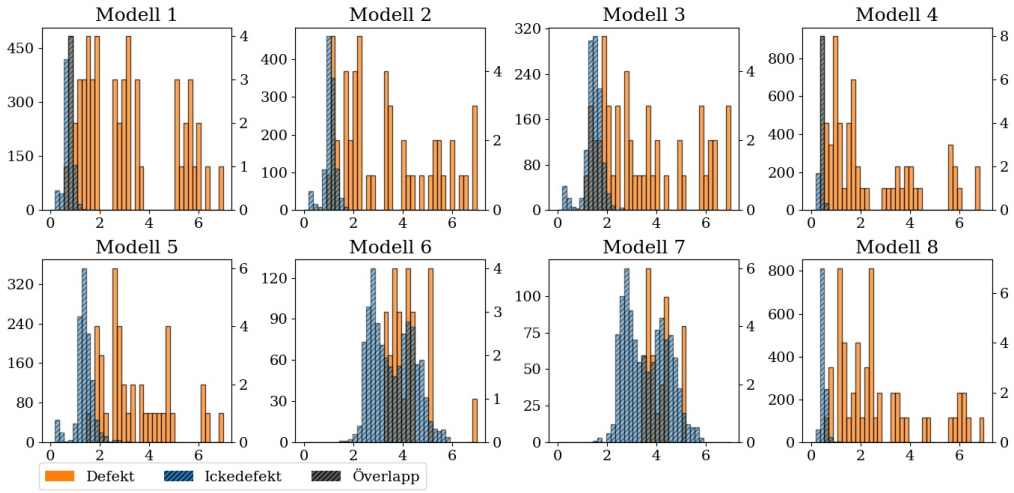
Figur 4.3: Ett urval av rekonstruerade kretssektioner som annoterats ickedefekta i Gen2 och varje rekonstruktions MMSE8 med det initiala CAE-nätverket, Figur 3.4. För varje bild visar en röd ruta var felet är som störst och alltså var MMSE8 har beräknats.

Det finns två specifika typer av defekter som ofta överlappar med ickedefekta testbilder i MMSE8 ($i = 27, \ldots, 32$ och $i = 60, \ldots, 65$ i Figur 4.2). De två typerna av defekter visas som de två första exemplen i Figur 4.4, där ett urval av rekonstruerade defekta kretssektioner visas. Notera att de defekta kretssektionerna $i = 27, \ldots, 32$ och $i = 60, \ldots, 65$ är två olika typer av defekter som har utökats genom translation, se avsnitt 3.4.1. Det observerades att de translaterade bilderna där defekten var placerad mer centralt rekonstruerades med högre MMSE8 än de bilder där defekten förekom vid bildens kant.

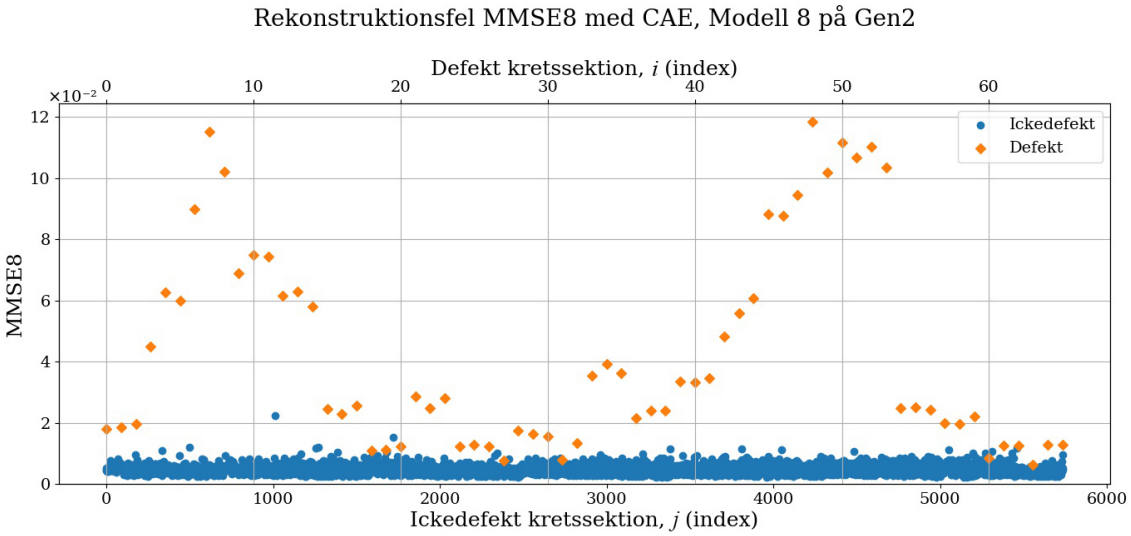


Figur 4.4: Urval av rekonstruerade defekta kretssektioner från Gen2 och varje rekonstruktions MMSE8 med det initiala CAE-nätverket, Figur 3.4. Notera att MMSE8 hittar de defekta sektionerna i samtliga bilder, vilka markeras med rödvita rutor.

I Figur 4.4 syns också tydligt att defekter med vita områden ger högre rekonstruktionsfel än svarta och grå defekter. Lägst MMSE8 ges med defekten som liknar ett svart hål och högst MMSE8 ges av de större vita defekterna. Från försöket att optimera den initiala modellen, Figur 3.4, togs histogram fram för att visa fördelningen av MMSE8 för defekta och ickedefekta kretssektioner för Modell 1-8 (Tabell 3.3). Histogrammen visas i Figur 4.5 överlappet i MMSE8 mellan defekta och ickedefekta kretssektioner är minst för Modell 8. Ett så litet överlapp som möjligt innebär att ett säkrare tröskelvärde kan bestämmas för att förutspå defekt eller ickedefekt kretssektion vid tillämpning av modellen på tidigare osedd data.



Figur 4.5: Histogram över MMSE8 för Modell 1-8. Höger och vänster y-axel motsvarar antal rekonstruktioner av defekta respektive ickedefekta kretssektioner för specifika värden på MMSE8. Av de ickedefekta kretssektionerna i testdatan används i bilden endast 20 % av dem (1147 stycken), vilket räcker för att ge en fullgod representation av fördelningen. Intervallet för MMSE8 har valts för att förtydliga överlappet i MMSE8 mellan defekta och ickedefekta kretssektioner, vilket visas i en mörkare grå nyans.



Figur 4.6: MMSE8 för samtliga defekta sektioner och de ickedefekta sektioner vid rekonstruktion med Modell 8. Index för defekta och ickedefekta testbilder visas på varsin x-axel som i respektive j.

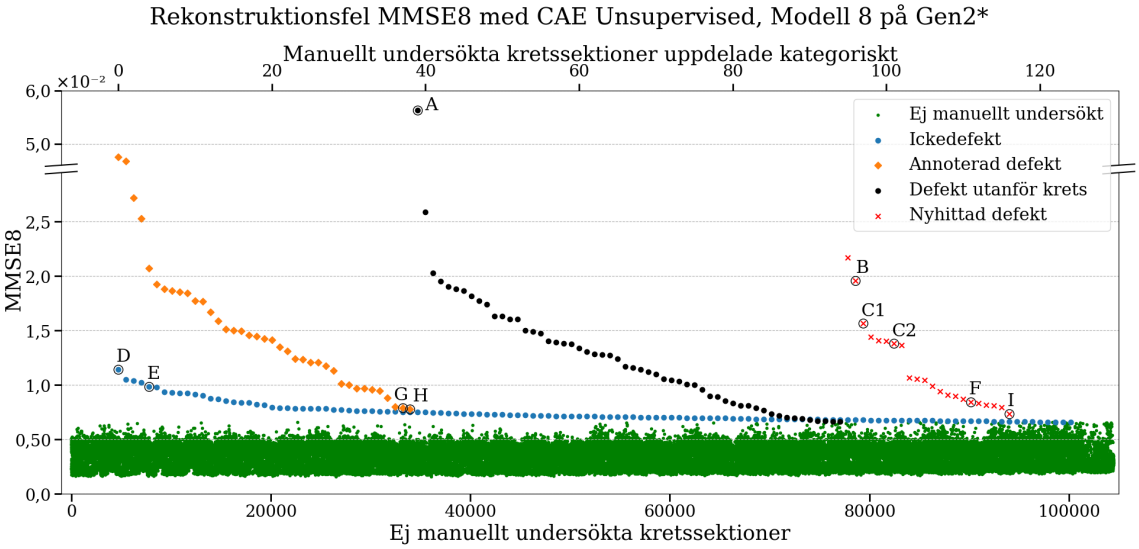
I Figur 4.6 syns att de defekter som hade lägst fel med den initiala modellen i Figur 4.2, nu har högre fel med Modell 8. De rekonstruktionsfel för defekta kretssektioner som fortfarande är låga ($i = 27,\ 31,\ 60,\ 63$) är alla exempel där defekten är placerad vid bildens kant. Att defekter som ligger vid kanten av bilderna ger lågt MMSE8 behöver dock inte vara något problem. Vid undersökning av en ny krets kan defekter i utkanten av kretssektioner direkt undvikas genom att styra hur tätt kretssektioner ska tas ut från SEM-bilden av kretsen.

Att träna Gen2 med Modell 8 i 100 epoker tog ungefär 20 minuter vid användning av PC-klustret Vera. Vidare var tidsåtgången ungefär en minut för att ta fram rekonstruerade bilder och beräkna MMSE8.

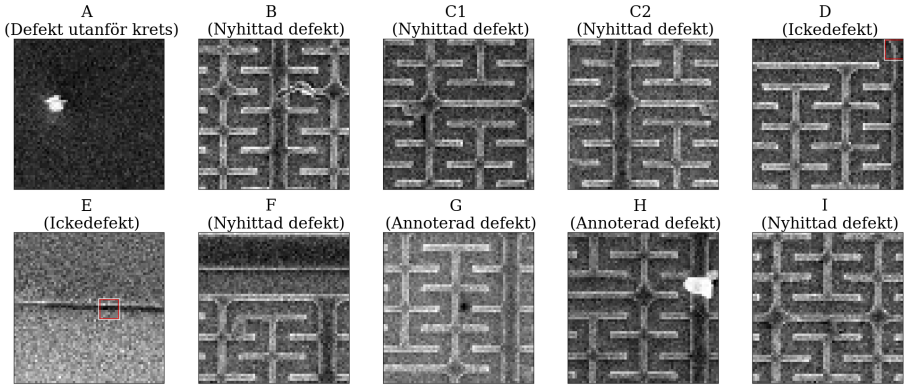
4.2.2 CAE – Unsupervised Learning

För Gen2* och Gen3 användes Unsupervised Learning med Modell 8. För Gen2* visas resultaten i Figur 4.7, där respektive kretssektions MMSE8 visas och utvalda kretssektioner är markerade med **A**-**I**. De 240 kretssektioner med högst MMSE8 i fallande ordning undersöktes manuellt och kategoriserades som *Ickedefekt*, *Annoterad defekt* (från Gen2), *Defekt utanför krets* och *Nyhittad defekt*. De annoterade defekterna är alltså samma som i den annoterade datamängden Gen2 och var alla hittade vid markering **G**. **H** är den sista av de undersökta kretssektionerna där en annoterad defekt förekom och som följd av överlappet i kretsuppdelning förekom defekten i **H** även i en annan kretssektion och där med högre MMSE8 på $1{,}2 \cdot 10^{-2}$. Det syns att före **H** har ungefär 30 ickedefekta kretssektioner högre MMSE8. I figuren syns även att det finns flera defekter utanför kretsen som modellen lyckas detektera. Modellen lyckades även hitta nya defekter som inte var annoterade i Gen2. Resterande kretssektioner som ej undersöktes manuellt syns som gröna punkter i figuren.

De utvalda kretssektionerna från Figur 4.7 syns i Figur 4.8. **A** visar ett exempel på en defekt utanför kretsen. **B** är en nyhittad defekt med högt MMSE8. Kretssektionerna **C1** och **C2** innehåller samma defekt, en konsekvens av kretssektionernas överlapp. Notera att **C2** har defekten på kanten och även lägre MMSE8 jämfört med **C1** där defekten är lokaliserad mer centralt. **D** och **E** är ickedefekta kretssektioner och i dem visas där Modell 8 identifierar MMSE8 med en rödvit ruta. I **D** markeras ett rundat hörn och i **E** en del av SEM-bilden där en annan del än kretsen har fångats. I kategorin *Ickedefekta* är enbart kretssektioner som liknar **D** och **E**, alltså rundade hörn och delar utanför kretsen. Framförallt är det rundade hörn som dominerar bland ickedefekter. Andra modeller med färre antal filter var inte lika känsliga för runda hörn, men gav samtidigt lägre MMSE8 för den mörka defekten som visas i **G**. **F** och **I** är nyhittade defekta sektioner med lågt MMSE8.



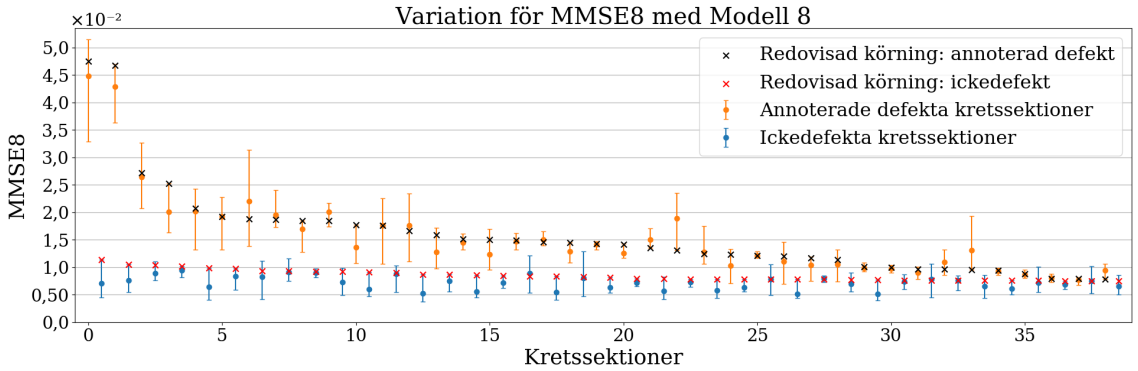
Figur 4.7: MMSE8 vid rekonstruktion med Modell 8 för samtliga kretssektioner i Gen2*. De 240 kretssektionerna med störst MMSE8 är manuellt undersökta och kategoriserade. Utvalda kretssektioner är markerade A-I och syns i Figur 4.8.



Figur 4.8: Markerade exempel på kretssektioner i Figur 4.7.

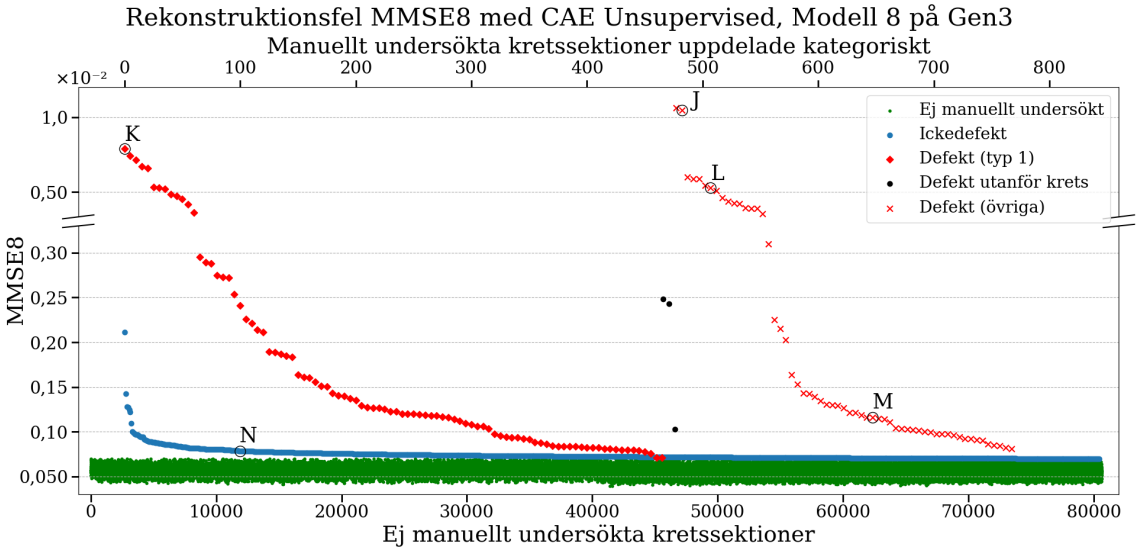
Reproducerbarheten för Modell 8 undersöktes genom att göra ytterligare 6 träningar med Modell 8. För varje körning slumpades ordningen för kretssektionerna och nätverkets initialvikter, vilket kan ge variation i resultatet. I Figur 4.9 syns spridningen och genomsnittet av MMSE8 för annoterade defekta kretssektioner och ickedefekta kretssektioner i

den redovisade träningen i Figur 4.7 och de ytterligare 6 träningarna. Spridningen är med avseende på minsta och största MMSE8 för respektive kretssektion och genomsnittet tas över samtliga träningar. En viss variation kan observeras, men överlag har de defekta kretssektionerna högre MMSE8 än de ickedefekta kretssektionerna. Notera att de ickedefekta kretssektionerna i figuren är de med störst MMSE8 för den redovisade körningen.



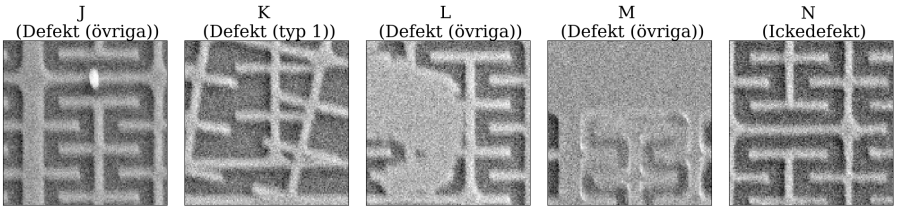
Figur 4.9: Genomsnitt och spridning av MMSE8 med Modell 8 på Gen2*, för utvalda defekta och ickedefekta kretssektioner. Sammanlagt 7 träningar redovisas där MMSE8 för resultatet i Figur 4.7 är markerade med kryss.

Resultatet för Gen3 visas i Figur 4.10. För Gen3 fanns ingen annoterad data, så de 1000 kretssektionerna med störst MMSE8 som undersöktes manuellt delades in i kategorierna *Ickedefekt*, *Defekt (typ 1)*, *Defekt utanför krets* och *Defekt (övriga)*. På samma sätt som ovan benämns de resterande kretssektionerna som *Ej manuellt undersökta*. De defekta sektionerna delades in i två kategorier då många kretssektioner innehöll typ 1-defekter (se Tabell A.1 för information om olika sorters defekter). Exempel på olika kretssektioner syns i Figur 4.11. Markerade kretssektionerna **J** - **N** i Figur 4.10 visas i Figur 4.11.



Figur 4.10: MMSE8 vid rekonstruktion med Modell 8 för samtliga kretssektioner i Gen3. De 1000 kretssektionerna med störst MMSE8 är manuellt undersökta och kategoriserade. Utvalda kretssektioner är markerade och visas i Figur 4.11. Alla defekta kretssektioner har plottats vid var femte punkt med avseende på övre x-axeln för att öka spridningen. Se Tabell A.1 i appendix för förklaring av defekttyperna.

Vid användning av PC-klustret tog det ungefär två timmar att träna Gen2* respektive Gen3 i 100 epoker. Fem minuter åtgick att köra datamängderna genom respektive modell samt ta fram MMSE8.



Figur 4.11: Utvalda Gen3-kretssektioner i Figur 4.10.

5 Diskussion

I detta avsnitt diskuteras framförallt det främsta resultatet i rapporten från avsnitt 4.2.2, där Unsupervised Learning tillämpades med CAE. Det beskrivs hur ett tröskelvärde för rekonstruktionsfelet kan användas för att klassificera nya kretssektioner, samt vilka defekter som var svårast att registrera med ett högt MMSE8. Vidare diskuteras eventuella framtida tillämpningar och nödvändiga verktyg för att reproducera resultaten i studien.

5.1 Unsupervised Learning med CAE

I Figur 4.7 och Figur 4.10 syns att rekonstruktionsfelet MMSE8 tydligt separerar majoriteten av defekta kretssektioner mot ickedefekta kretssektioner för Gen2* och Gen3 med Modell 8. Notera att Gen2* och Gen3 innehåller bilder som är helt obehandlade efter ha blivit fångade vid SEM. Modell 8 kräver alltså varken att kretssektionerna är annoterade (Novel Detection) eller förbehandlade för att möjliggöra klassificering med hjälp av ett tröskelvärde för rekonstruktionsfelet.

5.1.1 Framtagning av tröskelvärde och användning av CAE

Från resultatet med Modell 8 på Gen2*, Figur 4.7, kan ett tröskelvärde för MMSE8 exempelvis väljas som $6 \cdot 10^{-3}$, för att separera defekta och ickedefekta kretssektioner. För att visa skillnaden i tidsåtgång vid undersökning av kretsarna manuellt, jämfört med hjälp av en CAE, presenteras här ett exempel på en nytillkommen krets som liknar Gen2*. Det antas då att den nya kretsen ger ett dataset på 100000 bilder och att ungefär 300-500 bilder ger högre MMSE8 än det valda tröskelvärdet på $6 \cdot 10^{-3}$. Antagandet är i enighet med datasetet Gen2* och tillhörande resultat i Figur 4.7.

För en person med god kunskap om hur defekterna kan se ut uppskattas det att manuell undersökning av kretsarna i genomsnitt kräver 5 sekunder per bild, vilket ger ungefär 140 timmar för 100000 bilder. Om det antas att områden utanför kretsen, se Figur 3.3 **(b)**, utgör en femtedel av de totala bilderna och att de kan uteslutas vid manuell undersökning, kan tidsåtången reduceras till drygt 100 timmar. Framtagandet av MMSE8 för Gen2* med en färdigtränad Modell 8 tog ungefär en halvtimme på en vanlig arbetsdator, vilket antas gälla i detta exempel. Tröskelvärdet för MMSE8 ger då 300-500 bilder som bör inspekteras manuellt, vilket totalt resulterar i 1-2 timmar för att lokalisera defekterna.

Notera att vid manuell undersökning missades flera kritiska defekter som Modell 8 registrerade med högt MMSE8 för Gen2*, se Figur 4.7. Betrakta även Figur 4.8 och kretssektioner **F** och **I** med nyhittade defekter. Defekterna är otydliga och kräver viss ansträngning för att identifiera med det mänskliga ögat. Vidare kunde modellen detektera defekter av tidigare okänd art. Att ta fram tröskelvärdet kan även tänkas göras automatiskt. Rekonstruktionsfelet kan betraktas som normalfördelat med medelvärde ungefär mitt i brusgolvet. Genom att estimera en normalfördelning kan tröskelvärdet därefter väljas några standardavvikelser från medelvärdet.

5.1.2 Defekter med lägre rekonstruktionsfel

I resultatet, Figur 4.7 och 4.8, syns att den svarta defekten i Gen2* inte överstiger brusgolvet markant. Förklaringen till fenomenet är sannolikt att svarta defekter liknar mörkare områden som förekom på många kretssektioner. Modell 8 kan därför ha lärt sig indirekt att rekonstruera svarta defekter med lågt MMSE8. De vita defekterna hade även urskiljbart högre pixelintensitet än förekommande normala intensiteter på kretssektionerna.

Efter diskussion med handledarna tycktes de svarta defekterna vara någon form av organisk rest som har förkolnats. Om ett organiskt material undersöks med SEM riskerar det att brännas vid, vilket leder till förkolning. Exempel på när förbränning av organiska material sker som resultat av elektronmikroskopi kan observaras i [25]. Förkolning av en organisk defekt leder till att defekten svärtas i SEM-bilden och ser ut som ett svart “hål” i vågledaren. Organiska rester är inte en stark indikation om att en kortslutning har skett. De vita och grå defekterna kan exempelvis vara dammpartiklar eller plastrester från renrummet i MC2. De skulle även i värsta fall kunna ha en bit metall under sig, vilka är de farligaste typerna av defekter för mikrovågledarna. Vita och grå defekter är en större indikation om att en kortslutning gömmer sig under dem, och är av stort intresse att undersöka vidare. Se även Tabell A.1 för vidare information om defekterna.

5.2 Vidare optimering av modeller

Även om Modell 8 presterar mycket väl för de använda bilderna kan det begrundas hur nätverket kan göras ytterligare robust. I framtida fall kan det exempelvis förekomma mer komplexa nanostrukturer som kräver mer invecklade nätverk. Exempelvis kan det vara av intresse att undersöka tillskott av andra prestationsmetriker för testdata. Eftersom Modell 8 tycks vara mer känslig för vita defekter hade förslagsvis en metrik som är viktad mot mörkare pixelintensiteter kunnat utvecklas och användas utöver MMSE8. Tillskottet skulle potentiellt kunna leda till att det blir lättare särskilja svarta defekter från brusgolvet. Även optimering av antal epoker kan leda till bättre resultat, eftersom för lång träning kan tänkas leda till att brusgolvet nås för alla ickedefekta kretssektioner, och det mesta som finns kvar att optimera är de defekta sektionerna.

Resultatet på balanserad data i avsnitt 4.1 ger stöd åt en del av teorin som presenterades i avsnitt 2.4. DNN och CNN uppnådde högst maximal träffsäkerhet och medelträffsäkerhet vid träning under 100 epoker. Alltså tycks den ökade komplexiteten i modellerna leda till bättre prestation under samma antal epoker. Samtidigt misstänks TLPCA ha överanpassats, vilket föreslås av den låga träffsäkerheten och den märkbara skillnaden mellan

träffsäkerheten för träningsdatan och valideringsdatan i Figur 4.1.

5.3 Andra lösningar och framtida tillämpningar

Att testa andra modeller än CAE hade även varit intressant för obalanserad data. Exempelvis testades kort *Variational* Autoencoder (VAE), men avfärdades efter att CAE visade på goda resultat. VAE har med framgång tillämpats på flera olika typer av ML-problem som innefattar data i form av bilder [26]. VAE beräknar, utöver rekonstruktionsfel, även Kullback-Leibler-divergensen, som är ett mått på skillnaden mellan en estimerad och en förväntad sannolikhetsfördelning för det latenta lagret [27]. Risker med VAE innefattar att de kan vara svårare att träna utan att göra stora förenklingar eller antaganden [26].

För framtida tillämpningar är det viktigt att notera att tillämpning av CAE-nätverk för Unsupervised Anomaly Detection vilar på obalans mellan de två klasserna. En CAE tränad på en balanserad fördelning av defekta och ickedefekta kretssektioner hade sannolikt presterat mycket sämre, eftersom båda klasser då tränas ungefär lika mycket för att erhålla lågt rekonstruktionsfel. Om modellerna önskas reproduceras räcker det att ha en normal dator med en CUDA-kompatibel GPU. Träning med CPU är även rimligt för vissa av modellerna, såsom LR och DNN. PC-klustren på C3SE var nödvändiga först när flera fullständiga träningar krävdes för att optimera modellerna.

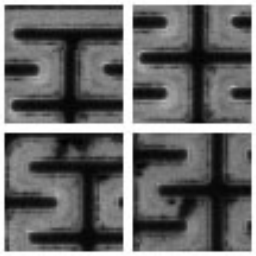
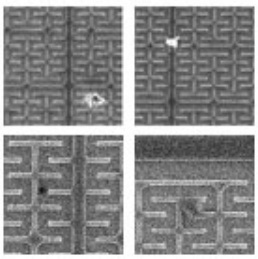
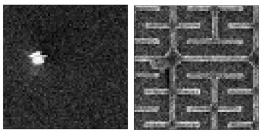
5.4 Slutsats

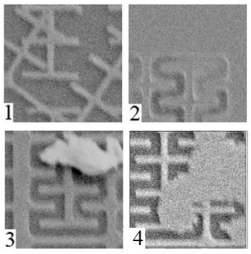
Denna studie har visat att det finns goda möjligheter att använda maskininlärning för att detektera defekter i mikrovågledarkretsar. Oavsett om ungefär lika många defekta sektioner som ickedefekta existerar eller om defekterna är underrepresenterade, finns ML-modeller som till övervägande del kan separera bilderna i deras respektive klasser. Den största bedriften med studien är att ha visat att Unsupervised Learning kan användas vid realistiska fall av mikrovågledarkretsar. Tillvägagångssättet kan bespara mycket tid genom att eliminera behovet att annotera kretssektionerna inför träning av en modell.

För mikrovågledarna på MC2 kan en CAE tränad med Unsupervised Learning eller Novel Detection urskilja defekta kretsar från ickedefekta genom den framtagna metriken MMSE8. Modellerna presterade väl för olika generationer av mikrovågledare med olika intensitet, kontrast, signal-brusförhållande och fraktalform. Generellt kan det därför tänkas att Unsupervised Learning med CAE bör fungera väl i andra fall för nanostrukturer som innehåller få defekter. Modellens hyperparametrar kan i så fall behöva optimeras för att lämpa sig för andra strukturer än mikrovågledarna på MC2. Med Modell 8 markeras tydligt förutspåelser om var defekterna ligger, utan behov av annoterade sektioner och utan förprocessering. Tidsåtgången för att lokalisera defekter med Modell 8 på en nytillkommen krets estimerades till att ta mellan 1-2 timmar. Det finns därför goda möjligheter till tidsbesparingar då manuell undersökning av samma krets uppskattades till att ta över 100 timmar. Sammanfattningsvis var därför studien lyckad och visar på en god framtid för maskininlärning inom nanotillverkning.

A Sammanfattning av använda datamängder

Tabell A.1: Beskrivning och bildexempel för dataseten Gen1, Gen2, Gen2* och Gen3 som användes i projektet. Filformatet för bilderna i samtliga dataset var TIFF. För ytterligare teori om mikrovågledarkretsarna hänvisas till A. Adamyan [1].

Dataset	Beskrivning	Bildexempel
Gen1	Gen1 bestod av totalt 2486 bilder i storleken 50×50 pixlar. Datasetet var annoterat med 1251 bilder på defekta kretssektioner, respektive 1235 på ickedefekta kretssektioner. Kretsen var av första generationens mikråvågledarkretsar på MC2 och innehöll ett stort område med defekter kallat *Macro Masking*. Macro Masking kan uppstå genom torkad *resist*[a] från elektronstålen och/eller vid misskötsel av kretsen. [a]En tunn polymerfilm som provytan täcks med	
Gen2	Gen2 tillhandahölls av handledarna som 9148 bilder i formatet 128×128, varav 22 bilder var av annoterade defekta kretssektioner. Kretssektionerna var från en krets av andra generationens mikrovågledarkretsar på MC2. Bilderna beskärdes till 64×64 för att bättre efterlikna Gen1 i storlek av kretsegenskaper i bilderna. I nästa kolumn visas exempel av bägge storlekar. Vanliga defekter i Gen2 syns som vita områden och kallas *Micro Masking*. Micro Masking uppstår då en dammpartikel i renrummet hamnar på kretsens resist från elektronstrålen efter tillverkning. Det förhindrar den supraledande filmen att appliceras under partikeln vid etsning och kan orsaka kortslutning mellan två fraktalextremiteter. Kortslutningen gömmer sig då under defekten. Gen2 innehåller även defekter som liknar svarta prickar, vilket anses vara organiska rester som har förkolnats.	
Gen2*	Gen2* var en uppdelning av samtliga SEM-bilder i råformat som använts för att skapa datasetet Gen2. Bilderna togs ut i storleken 64×64 med ett överlapp på 16 pixlar för varje bild, vilket resulterade i 104448 bilder. Förutom att fler kretssektioner erhölls gentemot Gen2, innehåller Gen2* även svarta bilder på områden utanför kretsen. I nästa kolumn visas ett exempel av Micro Masking på ett sådant område.	

Gen3	Gen3 bestod av 105 bilder i storleken 3072×2304 pixlar och var helt obehandlat efter bildupptagningen i renrummet. Den krets som användes här var av tredje generationens mikovågledarkretsar på MC2. Bilderna delades upp i storleken 128×128 med ett överlapp på 32 pixlar för varje bild, vilket resulterade i 80640 bilder. Defekterna i Gen3 var väsentligen av tre typer: 1) Dåligt fäste för resist. Kan orsakas av ett tunt lager vatten som hamnat på provytan som kretsen framställs på. 2) För låg exponering på elektronstrålen. 3) Micro Masking. 4) Polymerisering av resist från elektronstålen.	

B Pythonkod för CAE Unsupervised Modell 8

```
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D
from tensorflow.keras.models import Model
import numpy as np

# The size of the images
size = 64

# Loading images from the dataset
images = np.load('images.npy')

# Formatting and normalizing the images
images = images.astype('float32') / 255.
images = np.reshape(images, (len(images), size, size, 1))

# The architecture for the CAE, Model 8 follows below, in terms of encoder and decoder.

# Encoder:
input_img = Input(shape=(size, size, 1))
x = Conv2D(64, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

# 'encoded' is the latent layer and holds the compressed representation of the images

# Decoder:
x = Conv2D(512, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)

# Create the CAE model
```

```
autoencoder.compile(optimizer='adam', loss='mse')

# Train the model:
history = autoencoder.fit(images, images,
                          epochs=100,
                          batch_size=64,
                          shuffle=True,
                          validation_split=0.3,
                          )

# Reconstructs all images with the trained model
decoded_images = autoencoder.predict(images)
```

Referenser

[1] A. Adamyan, "Slow propagation line-based superconducting devices for quantum technology," Doktorsavhandling, Instutitionen för Mikroteknologi och Nanovetenskap, Chalmers tekniska högskola, Göteborg, 2016. [Online]. Tillgänglig: https://research.chalmers.se/en/publication/240894. [Hämtad: 2020-05-09].

[2] R. Chalapathy & S. Chawla, "Deep Learning for Anomaly Detection: A Survey," 24 jan 2019. [Online]. Tillgänglig: https://arxiv.org/abs/1901.03407v2. [Hämtad: 2020-02-05].

[3] T. Wuest, D. Weimer, C. Irgens & K. Thoben, "Machine learning in manufacturing: advantages, challenges, and applications," *Production & Manufacturing Research*, vol. 4, ss. 23-45, doi: 10.1080/21693277.2016.1192517.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York: Springer-Verlag, 2006.

[5] Stanford University. (2012). Machine Learning. [Online]. Tillgänglig: https://www.coursera.org/learn/machine-learning, https://www.youtube.com/playlist?list=PLLssT5z_DsK-h9vYZkQkYNWcItqhlRJLN. [Hämtad: 2020-04-26].

[6] P. J. L. Adeodato, G. C. Vasconcelos, A. L. Arnaud, R. A. F. Santos, R. C. L. V. Cunha & D. S. M. P. Monteiro "Neural Networks vs Logistic Regression: a Comparative Study on a Large Data Set," jan. 2004, Recife, Pernambuco, Brasilien. [Online]. Tillgänglig: https://www.researchgate.net/publication/220932293_Neural_Networks_vs_Logistic_Regression_a_Comparative_Study_on_a_Large_Data_Set. [Hämtad: 2020-02-14].

[7] D. M. Pelt & J. A. Sethian, "A mixed-scale dense convolutional neural network for image analysis," *Proceedings of the National Academy of Sciences of the United States of America*, jan. 9, 2018, ss. 254-259, doi: 10.1073/pnas.1715832114.

[8] I. Goodfellow, Y. Bengio & A. Courville, "Deep Learning," *MIT Press*, 2016. [Online]. Tillgänglig: https://www.deeplearningbook.org/. [Hämtad: 2020-05-05].

[9] K. He, X. Zhang, S. Ren & J. Sun, "Identity Mappings in Deep Residual Networks," *Microsoft Research*, 2016. [Online]. Tillgänglig: https://arxiv.org/pdf/1603.05027.pdf. [Hämtad: 2020-05-05].

[10] K. Leung & C. Leckie, "Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters," *Twenty-Eighth Australasian Computer Science Conference (ACSC2005), 2005, Newcastle, NSW, Australia, January/February 2005*, Vladimir

Estivill-Castro, upplaga 38, ss. 333-342. [Online]. Tillgänglig: https://crpit.scem.westernsydney.edu.au/confpapers/CRPITV38Leung.pdf. [Hämtad: 2020-04-01].

[11] M. Sakurada & T. Yairi, "Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction," *MLSDA'14: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, dec. 2014, ss. 4-11. doi: 10.1145/2689746.2689747.

[12] A. Krizhevsky, I. Sutskever & G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, F. Pereira, C.J.C. B. L. Bottou & K.Q. Weinberger.

[13] J. Brownlee, *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*, Machine Learning Mastery, 2018, 141-151. [Online]. Tillgänglig: Google Books.

[14] X. Glorot, A. Bordes, & Y. Bengio, "Deep sparse rectifier neural networks, *Journal of Machine Learning Research 15*" jan. 2010, vol. 15, ss. 315–323. Tillgänglig: https://www.researchgate.net/publication/215616967_Deep_Sparse_Rectifier_Neural_Networks. [Hämtad: 2020-04-09].

[15] A. Graves, "Adaptive Computation Time for Recurrent Neural Networks," mar. 2016. [Online]. Tillgänglig: https://arxiv.org/abs/1603.08983. [Hämtad: 2020-05-11].

[16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, & X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Tillgänglig: https://www.tensorflow.org/about. [Hämtad: 2020-02-13].

[17] C. François, "Keras," GitHub, 2015. [Online]. https://github.com/fchollet/keras. [Hämtad: 2020-02-13].

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, et al., "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, okt. 2012, ss. 2825–2830. [Online]. Programvara tillgänglig: https://scikit-learn.org/stable/. [Hämtad: 2020-02-13].

[19] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000. [Online]. Tillgänglig: https://opencv.org/. [Hämtad: 2020-02-14].

[20] Microsoft Corporation, “GitHub,” [Online]. Tillgänglig: https://github.com/. [Hämtad: 2020-04-17].

[21] L. Biewald, “Experiment Tracking with Weights and Biases,” 2000. [Online]. Programvara tillgänglig: https://wandb.com/

[22] A. LeNail, “NN-SVG: Publication-Ready Neural Network Architecture Schematics,” *Journal of Open Source Software*, 2019, vol. 4, no. 33, s. 747. [Online]. doi: 10.21105/joss.00747.

[23] D. P. Kingma and J. L. Ba, “Adam: a method for stochastic optimization,” *International Conference on Learning Representations (ICLR)*, 2015. [Online]. Tillgänglig: https://arxiv.org/pdf/1412.6980.pdf. [Hämtad: 2020-04-10].

[24] F. Chollet, “Building Autoencoders in Keras,” The Keras Blog, 14 maj 2016. [Online]. Tillgänglig: https://blog.keras.io/building-autoencoders-in-keras.html. [Hämtad: 2020-04-27].

[25] J. B. Park, Y. Kim, S. Kim, J. M. Yoo, Y. Kim, R. Gorbachev, I. I. Barbolina, S. J. Kim, S. Kang, M. Yoon, S. Cho, K. S. Novoselov & B. H. Hong, “Non-destructive electron microscopy imaging and analysis of biological samples with graphene coating,” *2D Materials*, vol. 3, no. 4, 28 sep. 2016. [Online]. doi: 10.1088/2053-1583/3/4/045004.

[26] C. Doersch, “Tutorial on Variational Autoencoders,” 16 aug. 2016. [Online]. Tillgänglig: https://arxiv.org/abs/1606.05908. [Hämtad: 2020-04-28].

[27] V. Prokhorov, E. Shareghi, Y. Li, M. T. Pilehvar & N. Collier, “On the Importance of the Kullback-Leibler Divergence Term in Variational Autoencoders for Text Generation,” *Association for Computational Linguistics*, Proceedings of the 3rd Workshop on Neural Generation and Translation, Hong Kong, nov. 2019, ss. 118-127. doi: 10.18653/v1/D19-5612.

INSTITUTIONEN FÖR MIKROTEKNOLOGI
OCH NANOVETENSKAP
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY