KU LEUVEN

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN

CAMPUS BRUGGE

Ontwikkeling en validatie van een detectiealgoritme voor oppervlaktefouten op vlakke platen

Karel DEBEDTS

Promotor: Prof. dr. ir. Verbeke M.
Co-promotor: dr. ing. De Ryck M.
ir. Desmet T.

Masterproef ingediend tot het behalen van de graad van master of Science in de industriële wetenschappen: elektronica-ICT

Academiejaar 2023 - 2024

Dankwoord

Ik wil mijn dankbaarheid uitdrukken aan iedereen die mij geholpen heeft in het proces van het realiseren van de master thesis. Allereerst wil ik mijn interne copromotor dr. Ing. Matthias De Ryck bedanken voor zijn uitgebreide tweewekelijkse opvolging en feedback. Vervolgens wil ik Tom Desmet van Decospan bedanken voor het aanleveren van de benodigde (financiële) data. Ook wil ik prof. Mathias Verbeke bedanken als promotor voor het realiseren van dit thesisonderwerp. Tenslotte wil ik mijn vrienden en familie bedanken voor het faciliteren van het proces.

Samenvatting

De kwaliteitscontrole aan het einde van een productielijn is cruciaal, aangezien deze de laatste gelegenheid biedt om defecte producten te identificeren voordat ze de consument bereiken. Deze thesis behandelt specifiek de kwaliteitscontroleprocessen bij Decospan, een producent van fineerplaten. Momenteel vertrouwt het bedrijf op handmatige inspecties door operators, die erin slagen om 70% van de defecte platen te identificeren. Echter, de resterende 30% resulteert in verhoogde kosten en verminderde klanttevredenheid. Deze masterproef onderzoekt de implementatie van machinevisie voor automatische foutdetectie, met een specifieke focus op het gebruik van deep learning via een convolutioneel neuraal netwerk (CNN). Het einddoel is het ontwikkelen van een model gebaseerd op de Decospan-dataset, met nadruk op zowel de prestaties als de inferentietijd.

De dataset wordt gekenmerkt door een beperkt aantal afbeeldingen en bijbehorende fouten, in combinatie met beelden van hoge resolutie. Gezien een CNN gewoonlijk een evenwichtige dataset met voldoende beelden in lage resolutie vereist, worden er specifieke methodologieën geïmplementeerd om aan deze vereisten te voldoen. Door de tijdsintensieve aard van het verzamelen van samples is een proof of concept ontwikkeld voor unsupervised learning, waarbij samples automatisch worden verzameld. Daarnaast wordt er gebruik gemaakt van pretrained modellen om de beperkte omvang van de dataset te compenseren. Verder worden augmentatietechnieken en samplingmethoden toegepast om elke foutcategorie adequaat te vertegenwoordigen. Om de resolutie van de samples te beperken maar detail te behouden, wordt de Slicing Aided Hyperinference (SAHI) methode toegepast. Dit onderzoek experimenteert vervolgens met verschillende netwerkarchitecturen en subsets van de dataset om optimale prestaties te bereiken.

De eerste focus ligt op detectie, waaruit blijkt dat een binaire CNN met een Resnet 18 backbone getraind op alle houtsoorten, de beste resultaten oplevert met een nauwkeurigheid van 98,9% en een recall van 75,9%. Vervolgens wordt overgegaan op classificatie, waarbij het Yolov8-model, getraind enkel op fouten in alle houtsoorten, een nauwkeurigheid van 72,7% bereikt. Daarna wordt een "2-stage approach"voorgesteld, waarbij een binaire CNN enkel defecte windows doorgeeft aan het Yolov8-netwerk. Deze benadering realiseert de laagste inferentietijd voor zowel detectie als classificatie. Geïmplementeerd op een embedded apparaat wordt een plaat in maximaal 2,75 seconden verwerkt. Tenslotte toont een financiële analyse aan dat het AI-systeem een jaarlijks rendement van 32,8% oplevert, met een terugverdientijd

van minder dan twee jaar.

Trefwoorden: Machinevisie, CNN, SAHI, ResNet, Yolo, Supervised learning

Summary

Quality control at the end of a production line is crucial, as it provides the last opportunity to identify defective products before they reach the consumer. This thesis specifically addresses the quality control processes at Decospan, a producer of veneer sheets. Currently, the company relies on manual inspections by operators, who manage to identify 70% of the defective sheets. However, the remaining 30% results in increased costs and reduced customer satisfaction. This master's thesis investigates the implementation of machine vision for automatic fault detection, with a specific focus on the use of deep learning via a convolutional neural network (CNN). The goal is to develop a model based on the Decospan dataset, emphasizing both performance and inference time.

The dataset is characterized by a limited number of images and associated errors, combined with high-resolution images. Given that a CNN typically requires a balanced dataset with sufficient examples in low resolution, specific methodologies are implemented to meet these requirements. Due to the time-intensive nature of sample collection, a proof of concept for unsupervised learning is developed, where samples are automatically collected. Additionally, the use of pretrained models compensates for the limited size of the dataset. Furthermore, augmentation techniques and sampling methods are applied to adequately represent each error category. To limit the resolution of the samples but retain detail, the Slicing Aided Hyperinference (SAHI) method is used. This research then experiments with different network architectures and subsets of the dataset to achieve optimal performance.

The initial focus is on detection, which shows that a binary CNN with a Resnet 18 backbone trained on all types of wood delivers the best results with an accuracy of 98.9% and a recall of 75.9%. Subsequently, the focus shifts to classification, where the Yolov8 model, trained only on errors in all wood types, achieves an accuracy of 72.7%. A "2-stage approachïs then proposed, where a binary CNN only passes defective windows to the Yolov8 network. This approach realizes the lowest inference time for both detection and classification. Implemented on an embedded device, a sheet is processed in a maximum of 2.75 seconds. Finally, a financial analysis shows that the AI system yields an annual return of 32.8%, with a payback period of less than two years.

Keywords: Machine vision, CNN, SAHI, ResNet, Yolo, Supervised learning

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Lijst van afkortingen

AI	Artificial Intelligence
ANN	Artificial Neural Network
AUROC	Area Under the Receiver Operating Characteristic
AP	Average Precision
CNN	Convolutional Neural Network
DETR	Detection Transformer
DORI	Detection, Observation, Recognition, and Identification
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
MAP	Mean Average Precision
MV4QC	Machine Vision for Quality Control
NAS	Neural Architecture Search
PCA	Principal Component Analysis
PLC	Programmable Logic Controller
RESNET	Residual Network
ROI	Return on Investment
RNN	Recurrent Neural Network
SAHI	Slicing Aided Hyper Inference
SVM	Support Vector Machine
VLAIO	Vlaams Agentschap Innoveren & Ondernemen
YOLO	You Only Look Once

Hoofdstuk 1

Inleiding

In deze scriptie wordt het potentieel van machinevisie onderzocht en geëvalueerd als een innovatieve oplossing voor het verbeteren van het kwaliteitscontroleproces bij Decospan, een bedrijf uit Menen dat fineer maakt voor interieurontwerp. Het gebruik van machinevisie in de context van fineerproductie biedt de mogelijkheid om nauwkeurige en efficiënte inspecties uit te voeren, wat kan leiden tot verbeterde productkwaliteit en kostenbesparingen. Dit onderzoek maakt deel uit van het VLAIO Technology Transfer (Tetra) project, genaamd "Machine vision for quality control" (MV4QC) uitgevoerd door KU Leuven Brugge en Hogeschool VIVES Kortrijk. Het doel hiervan is het vertalen van recent beschikbare kennis naar concrete info of toepassing voor de industrie. MV4QC wilt de implementatie van machinevisietechnieken voor kwaliteitscontroles bij ondernemingen versnellen door de beschikbare kennis en innovatieve technologie hieromtrent samen te brengen, te structureren, en op een gangbare en pragmatische manier aan te reiken aan de ondernemingen uit de doelgroep.

1.1 Thesis context

Decospan is gespecialiseerd in het vervaardigen van fineer, een dunne houtlaag die ontstaat door het snijden van houten stammen met een mes. Deze methode vermijdt zaagverliezen, wat betekent dat elk stuk hout optimaal wordt benut. De fineervellen, afkomstig van dezelfde stam, worden samengehouden om een consistente afwerking te waarborgen en vinden hun toepassing in diverse producten zoals meubelpanelen, wandbekleding en parket. Decospan biedt een uitgebreid assortiment van meer dan 150 verschillende houtsoorten aan, waarbij diverse snijtechnieken worden toegepast om specifieke visuele kenmerken te verkrijgen. Aangezien elke boom uniek is, zijn geen twee panelen identiek. Omwille van die reden hanteert het bedrijf een graderingssysteem voor fineer, waarbij letters de kwaliteit aanduiden. Een A+ fineerstapel word beschouwd als de hoogste kwaliteit, terwijl een C- de minst kwalitatieve optie vertegenwoordigt, met mogelijke beperkte imperfecties in de lagere klassen. Klanten hebben de flexibiliteit om de voorkant van een paneel te bekleden met een hogere kwaliteit fineer, terwijl ze een achterkant kunnen kiezen met een lager kwaliteitsniveau. Dit wordt vaak

gebruikt bij kastdeuren, waar de binnenkant esthetisch minder belangrijk is. De verwachting is dat de implementatie van machinevisie bij Decospan niet alleen zal leiden tot verbeterde productkwaliteit, maar ook tot een verhoogde operationele efficiëntie.

1.2 Probleemstelling

De schuurlijn vormt de laatste essentiële stap in het productieproces van Decospan. Deze fase is cruciaal voor het verfijnen en perfectioneren van de fineerproducten voordat ze de markt op gaan. Tijdens deze stap worden de fineervellen zorgvuldig geschuurd om een gladde en gelijkmatige textuur te verkrijgen. Deze procedure is van groot belang om eventuele oneffenheden, ruwheid of ongewenste kenmerken te verwijderen die tijdens eerdere stadia van het productieproces zijn ontstaan. De schuurlijn vertegenwoordigt niet alleen de laatste fase van het productieproces bij Decospan, maar het is tevens de laatste mogelijkheid om eventuele onvolkomenheden te identificeren voordat het fineer aan de klant wordt geleverd. Hierdoor is kwaliteitscontrole in deze fase van cruciaal belang. Er is dus nood aan een systeem dat fouten kan analyseren in deze productielijn, zonder de productielijn te vertragen, maar toch met hoge nauwkeurigheid voorspellingen kan doen. Momenteel schakelt Decospan hiervoor een lijnoperator in. Hoewel zijn hoofdtaak het bedienen van de machine is, wordt verwacht dat deze persoon ook kwaliteitscontrole uitvoert. Echter verheldert een verdere analyse door Decospan dat deze taak implicaties met zich meebrengt: de lijn verplaatst platen met een hoge snelheid en oppervlaktefouten zijn relatief klein. Hierdoor is het niet evident voor de manuele operator om dit met het blote oog eruit te halen. Vooral subtiele fouten verlaten de productie naar de klant en resulteren vervolgens in dure claims. In volgende hoofdstukken wordt de rol van machinevisie onderzocht om een systeem te creëren en implementeren dat een oplossing biedt voor deze uitdaging.

1.3 Doelstelling

Het globale doel van deze thesis is het ontwikkelen en valideren van een visueel detectie systeem om defecten in vlakke houten platen te detecteren en te classificeren. Hierbij heeft Decospan 2 cruciale vereisten vooropgesteld:

- De accuracy per klasse, deze moet 90 percent of meer zijn
- De snelheid, deze moet binnen de 3 seconden een voorspelling doen

Dit wordt verder opgesplitst in concrete onderzoeksvragen, voortvloeiend uit de problematiek van Decospan. De eerste onderzoeksvragen richten zich op de vergelijking van twee methodologische benaderingen: de eerste benadering omvat het implementeren van een state-of-the-art model op de gehele dataset, een benadering die vandaag de dag in de industrie gebruikt wordt. De tweede benadering stelt een tweedelig proces voor waarbij eerst fouten uit de data

worden geëxtraheerd om vervolgens een gespecialiseerd model te trainen dat uitsluitend op deze foutieve data is gericht. Als laatste is er een onderzoeksvraag die peilt naar de mogelijke methoden om afbeeldingen, afkomstig van de Decospan dataset, te voeden aan het model.

- Kan de gewenste performantie bekomen worden met een "state-of-the-art" multi-class (supervised) detectiemodel, toegepast op alle data?
- Kan een (unsupervised) anomaliedetectie model gebruikt worden om de defecten af te zonderen?
- Kan een (supervised) binaire classifier gebruikt worden om de defecten af te zonderen en wat is de performantie tegenover anomaliedetectie?
- Kan een (supervised) multi-class classificatie model dat getraind is op enkel defecte data betere resultaten bieden dan een model getraind op alle data?
- Biedt feature selection een hogere model performantie voor de Decospan dataset?
- Zorgt het trainen van een model per houtsoort voor een hogere performantie in vergelijking met een model dat geldt voor alle houtsoorten?
- Welke methoden zijn effectief voor het voeden van een beperkt aantal hoge-resolutie afbeeldingen uit de Decospan dataset in een model, met inachtneming van real-time verwerkingsbeperkingen?

Hoofdstuk 2

Literatuurstudie

2.1 Mogelijke oplossingsmethoden

Zoals vermeld in de titel van de thesis, wordt er gekeken naar machinevisie. Er zijn meerdere manieren om anomalieën te detecteren op vlakke houten platen. Het onderzoek van Shi et al. [1] vat relevante methoden samen met de bijhorende voor- en nadelen: air-coupled ultrasonic technology, stress wave-technologie, 3D lasertechnologie, computed tomography-based X-ray, colorimetrische technieken en deep learning.

Een eerste mogelijkheid is het gebruik van air-coupled ultrasonic technology. Dit is een ultrasone meettechniek die defecten intern in het hout kan detecteren op basis van densiteit variaties in het hout. Deze methode is zeer gevoelig aan variaties van de omgeving en is dus niet stabiel genoeg voor industriële toepassingen. Een volgende aangehaalde oplossing, is het inzetten van stress wave-technologie. Deze technologie kan de grootte en vorm van interne defecten detecteren op basis van gesegmenteerde voortplantingsstralen van stressgolven. De verwerving van stressgolfsignalen vereist echter dat de sensoren stevig aan het oppervlak van het hout zijn bevestigd en de afstand tussen sensoren moet vaststaan. Nog een methode is het gebruik van 3D lasertechnologie. Een laser kan accuraat de vorm van houten platen opmeten. Echter is de laser niet precies genoeg om kleine defecten te detecteren. Een veelbelovende techniek is het gebruik van computed tomography-based X-ray. Dit wordt ook door Shi et al [1] besproken. Met op computertomografie gebaseerde röntgenstraling worden knoppen gescheiden op basis van dichtheid en vochtigheid. Hoewel een zeer grote accuraatheid gehaald wordt en de technologie als betrouwbaar beschouwd wordt, blijft röntgenstraling schadelijk voor de mens. Vervolgens onderzochten Shi et al. [1] colorimetrische technieken: een klassieke niet data-gedreven machinevisie. In dit onderzoek werd de afbeelding gemanipuleerd met behulp van Hue-Saturation-Intensity kleurmodellen in combinatie met mediane vervaging, morfologische operatoren en de watershed-transformatie. Het probleem met deze bewerkingen is echter dat ze specifiek geoptimaliseerd zijn voor een bepaalde houtsoort en testomgeving. Zodra er andere kleurvariaties optreden, bijvoorbeeld door een andere houtsoort of testomgeving, blijkt deze methode niet meer robuust te zijn. Uiteindelijk wordt in

[1] een technologie behandeld die gebruik maakt van zowel computervisie als deep learning. Deze technologie wint aan populariteit, voornamelijk dankzij de opkomst van krachtige Graphics Processing Units (GPUs). Deep learning in machine vision verwijst naar het gebruik van diepe neurale netwerken voor het interpreteren van visuele informatie. Deze neurale netwerken, zoals Convolutional Neural Networks (CNNs), kunnen automatisch complexe patronen in beelden herkennen zonder handmatige functie-engineering.

2.2 Computervisie

De komende hoofdstukken richten zich op de literatuurstudie van computervisie en de uitbreiding van deep learning. Deze beslissing steunt op het werk van Shit et al. [1] omwille van hun argumenten die het potentieel en de praktische haalbaarheid van deze technologie onderstrepen. Zoals aangegeven door Shi et al. [1], heeft computervisie bewezen praktisch implementeerbaar te zijn met voordelen zoals lagere kosten, hogere accuraatheid en verbeterde veiligheid vergeleken met andere methoden. In dat opzicht is het gerechtvaardigd te besluiten dat deze piste het meeste kans biedt op het beantwoorden van de onderzoeksvragen. Ondanks het overvloedige aantal bronnen over dit onderzoekstopic, is er specifiek gezocht naar literatuur die van toegevoegde waarde is voor de praktische uitvoering van de probleemstelling. Dit met de globale topics van de onderzoeksvragen in het achterhoofd. Daarom wordt gestart met een literatuurstudie over unsupervised en supervised learning. Vervolgens wordt het invoeren van afbeeldingen in een neuraal netwerk, met een specifieke focus op de karakteristieken van de Decospan dataset, onderzocht. Er wordt aandacht besteed aan de uitdagingen die voortkomen uit de hoge resolutie van de afbeeldingen en de imbalanced verdeling van de dataset. Tenslotte wordt onderzocht hoe deze afbeeldingen efficiënt kunnen worden verwerkt binnen de grenzen van real-time prestatievereisten.

2.2.1 Unsupervised learning

Vooraleer ingezet wordt op supervised deep learning, raden Cohn et al. [2] aan om te kijken naar unsupervised technieken. Deze hebben significante opportuniteiten: omdat er geen data gelabled moet worden, kan de arbeidsintensieve stap van het creëren van een dataset overgeslaan worden, wat het interessant maakt voor de industrie. Cohn et al. [2] duiden dat voor het creëren van de ImageNet dataset 1,2 miljoen gelabelde images verzameld zijn, wat voor de industrie in praktijk een hoge kost met zich meebrengt en hierdoor beperkt haalbaar is.

2.2.1.1 K-means clustering

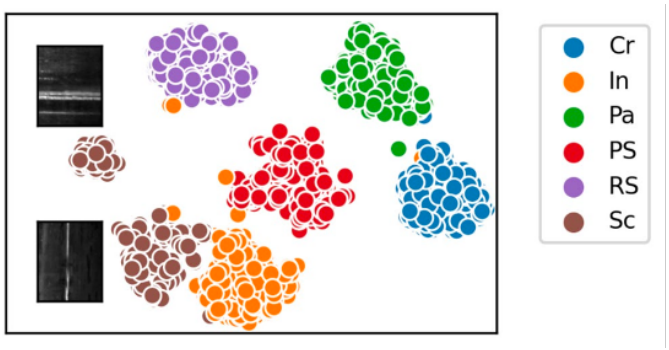
Als proof of concept hebben Cohn et al. [2] de K-means clustering techniek toegepast op de NEU surface defect database, welke staaldefecten catalogiseert. K-means clustering is

een unsupervised leermethode die tot doel heeft observaties te verdelen in vooraf bepaalde clusters, waarbij elke cluster wordt gedefinieerd door het gemiddelde (of 'centroid') van de punten die tot dat cluster behoren. Deze methode initieert een initiële set van k centroids, waarna het iteratief de toewijzing van elk datapunt aan het dichtstbijzijnde centroid verfijnt. Dit proces wordt voortgezet totdat er een convergentie is bereikt, waarbij de totale variatie geminimaliseerd wordt, wat resulteert in een verdeling van de dataset in discrete, homogene clusters. Cohn et al. [2] gebruiken hiervoor een methodologie bestaande uit drie delen:

1. Preprocessing
2. Feature extraction
3. Clustering

Een eerste stap is preprocessing. Hiervoor gebruiken Cohn et al. [2] histogram equalisation, om een uniforme belichting te hebben doorheen de samples. Dit omdat de belichting gevoelig kan zijn voor schommelingen in de omgeving of het aantal aanwezige defecten. Het zal de belichting van de afbeeldingen normaliseren, zo zijn er minder verschillen tussen de donkere en lichtere regio's. Ook wordt er extra contrast voorzien, waardoor bepaalde features meer opvallen.
Vervolgens dienen de features van de staalafbeeldingen geëxtraheerd te worden. Hiervoor gebruiken Cohn et al. [2] pre-trained neurale netwerken. De gewichten worden niet aangepast, enkel de waarden van bepaalde lagen worden geëxtraheerd. Op basis van voorgaande onderzoeken kwamen Cohn et al. [2] tot de conclusie dat het VGG16 netwerk het meest geschikt is voor het extraheren van features uit staal. Er worden lagen gekozen waar er informatie uit gehaald wordt: shallow layers zullen meer informatie geven over fijnere texturen, terwijl deeper layers meer info geven over grovere texturen. Op empirische wijze werd er tot de conclusie gekomen dat de eerste fully connected layer van vgg16 de beste performantie geeft. Echter bevat deze output maximaal 3,2 miljoen elementen, omwille van ruis en zero elements, ten gevolge van filters die niet activeerden. Omdat deze data een hoge dimensie heeft wordt principal component analysis (PCA) ingezet om de dimensionaliteit van de data te reduceren. PCA start met het berekenen van de covariantiematrix of de correlatiematrix van de data, om de onderlinge relaties tussen de variabelen te begrijpen. Vervolgens wordt er een eigenwaarde-decompositie uitgevoerd op deze matrix om de eigenvectoren en eigenwaarden te vinden. De eigenvectoren (die de richting van de principale componenten aangeven) worden dan gebruikt om de data te transformeren naar een nieuw coördinatensysteem, terwijl de eigenwaarden (die de magnitude aangeven) helpen bij het bepalen van de belangrijkheid van elke principale component. Vervolgens implementeren Cohn et al. [2] k-means clustering. Het resultaat hiervan is terug te vinden in figuur 2.1, wat illustreert dat de clustering methode veelbelovend oogt in het onderscheiden van de verschillende klassen. Hierbij werd een gemiddelde accuracy én recall van 99,5% gehaald.

Hoewel deze methode veelbelovend is, wordt duidelijk dat er een breed aanbod aan hyperparameters is dat manueel getuned moet worden. Zo dient er een goede backbone gekozen te



Figuur 2.1: K-means clustering voorbeeld op staal dataset [2]

worden in combinatie met een goede selectie van lagen. Tezamen met de dimensionaliteit reductie is dit een proces van trial en error, wat tijdsintensief kan zijn en ëxpert knowledge" kan vereisen.

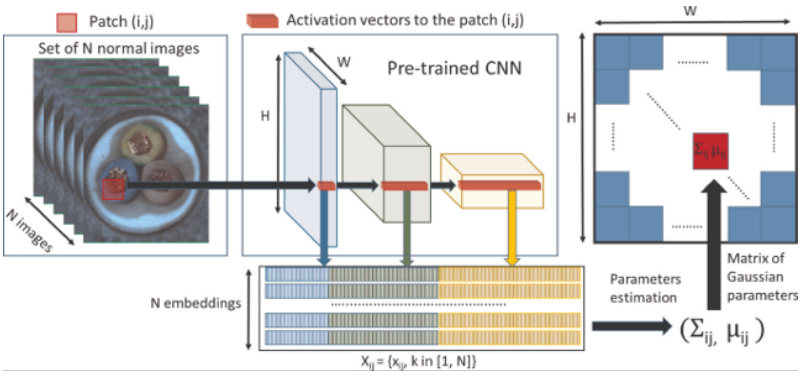
2.2.1.2 Anomaliedetectie

Het onderzoek van Defard et al. [3] concentreert zich op de detectie van anomalieën. Deze wordt omschreven als "het identificeren van onverwachte patronen of afwijkende kenmerken binnen een dataset die overwegend homogeen en natuurlijk is". Uit deze definitie valt op te maken dat het proces neerkomt op een vorm van binaire classificatie, waarbij een sample als normaal óf als afwijkend wordt beschouwd. In de industriële context wordt standaard de voorkeur gegeven aan supervised learning technieken, met name Convolutionele Neurale Netwerken (CNN's), voor classificatietaken. Echter, ook Defard et al. [3] bevestigen de voordelen van unsupervised learning en geloven dat de supervised - deep learning methode in de praktijk niet altijd de meest effectieve oplossing biedt:

1. Voor een operator is het moeilijk om foutieve samples te labelen en dus te detecteren
2. Het aantal foutieve samples is zeer beperkt, terwijl een CNN een groot aantal foutieve samples nodig heeft
3. Nieuwe samples van anomalieën kunnen onverwachtse patronen hebben, waardoor reeds getrainde CNN's minder goed presteren

Daarom wordt een nieuwe manier van training voorgesteld: de training set bevat enkel samples van de normale klasse. Hierdoor leert het netwerk de normale verdeling op basis van background samples, die in praktijk in overvloed aanwezig zijn. Het netwerk wordt dan geevalueerd door negatieve samples te voeden, waar het getraind model dan een afwijking kan waarnemen ten opzichte van de normale verdeling. In hun onderzoek passen Defard et al. [3] een model toe dat bekend staat als Patch Distribution Modeling (PaDiM). Deze methode begint met het opdelen van de invoersample in meerdere patches. Vervolgens wordt een

vooraf getraind Convolutioneel Neuraal Netwerk (CNN) ingezet om relevante kenmerken uit deze patches te extraheren. Met deze kenmerken wordt een Gaussische distributie berekend, wat de basis vormt voor de detectie van anomalieën, zichtbaar in figuur 2.2.



Figuur 2.2: PaDiM architectuur [3]

Defard et al. [3] rapporteren dat hun methode veelbelovend presteert ten opzichte van hedendaagse, geavanceerde Convolutionele Neurale Netwerken (CNN's). Dit met een aanzienlijk lagere tijds- en ruimtecomplexiteit, die bovendien niet toeneemt met de grootte van de dataset. Deze bevindingen werden gevalideerd met behulp van de MVTec Anomaly Detection (AD) dataset [4], een toonaangevende dataset ontworpen voor het evalueren van modellen die via one-class learning anomalieën detecteren. Hiervoor gebruikt men een metric genaamd "Area under the receiver operating charactersitic" (AUROC). Dit is een maatstaf die gebruikt wordt om de prestaties van een binair classificatiemodel te beoordelen. Het omvat verschillende metrics zoals de true positive rate en false positive rate bij verschillende drempelwaarden. Het beste resultaat wordt waargenomen met PaDiM en een efficientNetB5 backbone: een AUROC van 97.0%.

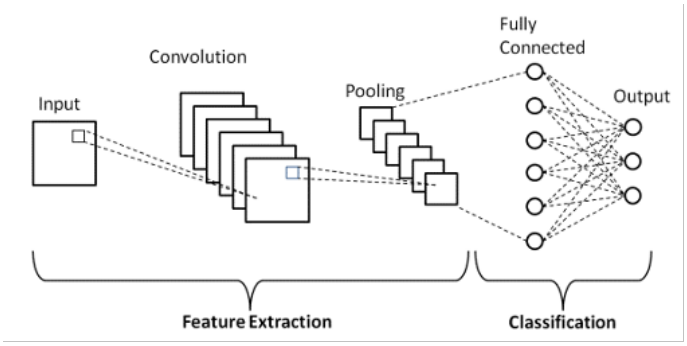
2.2.2 Supervised learning

Hoewel unsupervised technieken veelbelovend zijn, blijft supervised learning in combinatie met deep learning de meestgebruikte methode voor complexe classificatieproblemen met afbeeldingen. Dit berust op het principe van neurale netwerken: flexibele en krachtige modellen die een breed scala aan taken kunnen uitvoeren. Afhankelijk van de architectuur zijn er meerdere functionaliteiten mogelijk: bijvoorbeeld een objectiedetectie (segmentatie) segment of een object classificatie segment.

2.2.2.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNNs) behoren tot de categorie neurale netwerken en zijn ontworpen om automatisch en adaptief patronen te leren en te herkennen in data. Girshick et al. [5] verklaren dat de architectuur inspiratie vond bij de biologie: het menselijk brein.

Dit wordt weergegeven in figuur 2.3.



Figuur 2.3: CNN architectuur [6]

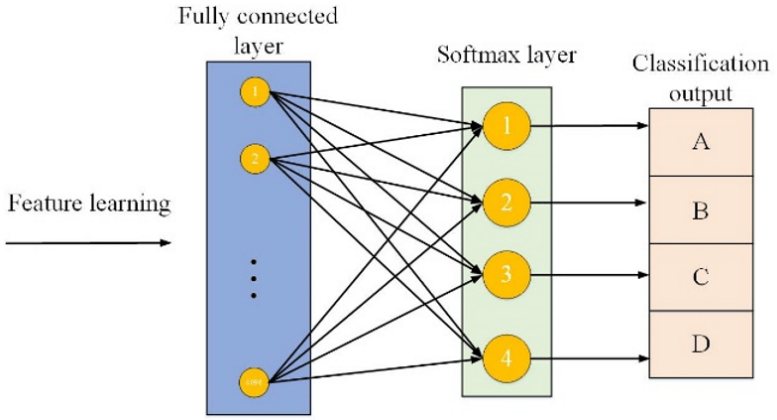
Convolutie: Het kernconcept achter CNN's is convolutie. Convolutie omvat het doorlopen van een filter (ook wel een 'kernel' genoemd) over het invoerbeeld. Dit filter detecteert bepaalde kenmerken, zoals randen of texturen. Door de convolutie worden deze kenmerken geëxtraheerd uit de invoer.
Activation functions: Na elke convolutielaag wordt meestal een niet-lineaire activatiefunctie toegepast, zoals de ReLU (Rectified Linear Unit). Deze functie introduceert niet-lineariteit in het model en helpt bij het leren van complexere patronen. Pooling: Na convolutie volgt een poolinglaag. Deze laag voorziet een dimensionaliteitsreductie.
Fully connected layers: Deze lagen functioneren als klassieke neurale netwerken en nemen de kenmerken, geleerd door de convolutie- en poolinglagen, mee om de uiteindelijke classificatie uit te voeren.

Classificatiemodellen:

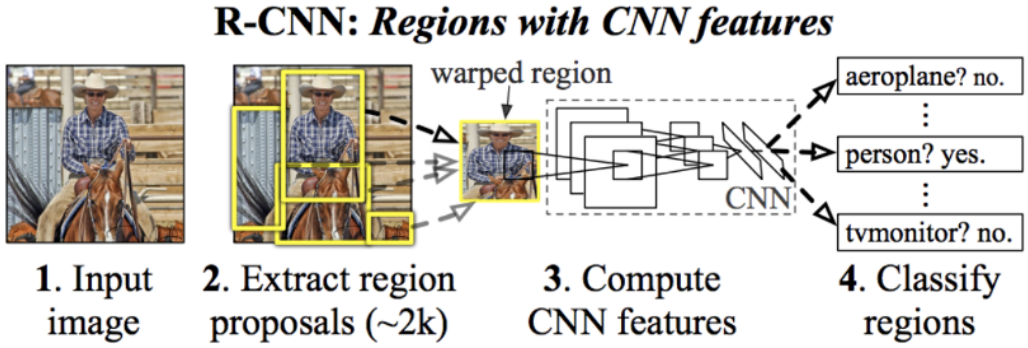
Classificatiemodellen bouwen voort op de architectuur van neurale netwerken, waarbij het doel is om invoerdata correct te classificeren in verschillende categorieën of klassen. Doorgaans wordt er aan de uitvoer van de CNN een softmax-laag geïmplementeerd, die de kans berekent van de waarschijnlijkheid dat het invoerbeeld tot een van de klassen behoort. De klasse met de hoogste waarschijnlijkheid wordt vervolgens gekozen als de voorspelde classificatie. Ter illustratie figuur 2.4.

Objectdetectie met CNN's:

In het onderzoek van Girshick et al. [5] wordt geduid dat CNN's succesvol blijken voor classificatietaken, maar tekortschieten op vlak van objectdetectie of segementatie. Daarom stellen ze een nieuw algoritme voor genaamd R-CNN: Region-based convolutional neural network. Dit model, weergegeven in figuur 2.7, combineert de kracht van een CNN met een region-based benadering.



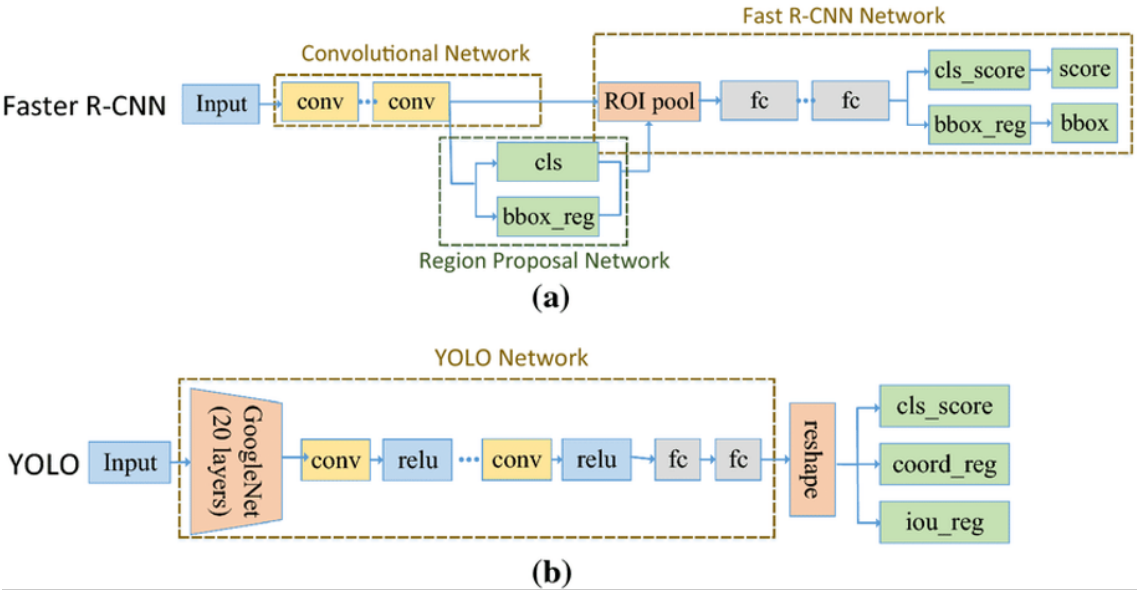
Figuur 2.4: CNN classificatie architectuur [7]



Figuur 2.5: R-CNN Architectuur [5]

R-CNN start met het verdelen van de afbeelding in meerdere regionen. Deze regionen zullen in de "region proposal stap", door middel van selective search, omgezet worden in een set van potentiële regionen die hoogstwaarschijnlijk objecten bevatten. Na het voorstellen van de regionen worden deze omgezet naar een input size dat het CNN verwacht. Hoewel deze techniek veelbelovend is, vraagt het algoritme een computationele kracht die aanzienlijk hoger is dan de klassieke niet data gedreven machinevisie. Dit resulteert in een hoge interferentie tijd.

Om deze problematiek aan te pakken introduceert het onderzoek van Akyon et al. [8] een nieuwe variant: single-stage neurale netwerken. Deze voeren objectdetectie en classificatie uit in één enkele stap. Een voorbeeld van zo'n netwerk is het You Only Look Once (YOLO) algoritme. YOLO verdeelt het invoerbeeld in een rooster en voert direct objectdetectie en classificatie uit op elk roosterpunt. Deze aanpak biedt het voordeel van snelheid, wat van cruciaal belang kan zijn in real-time toepassingen. Figuur 2.6 toont het verschil tussen een single- en multi-stagenetwerk.



Figuur 2.6: Vergelijking tussen R-CNN and Yolo architectuur [9]

2.2.2.2 Vision Transformer Networks (ViT's)

Een nieuw soort neuraal netwerk is een Vision Transformer (ViTs). Deze zijn gebaseerd op transormer architecturen, vandaag de dag gebruikt in bekende modellen zoals GPT-3 en GPT-4 voor natural language processing. Deze architectuur werd in 2021 voorgesteld door Dostrovitsky et al. [10] Een beeld wordt opgedeeld in meerdere patches, die vervolgens worden afgevlakt en door een reeks transformer-blokken worden verwerkt. Deze blokken gebruiken self-attention mechanismen om de relaties tussen alle patches te leren, waardoor het model globale context kan begrijpen zonder te vertrouwen op de lokale ruimtelijke hiërarchie die door convolutional lagen wordt vastgelegd. De kern van de transformer-architectuur binnen een ViT is het self-attention mechanisme. Dit mechanisme maakt het mogelijk voor het model om de relaties tussen alle patches van een afbeelding te wegen en te leren. In essentie berekent het voor elke patch een gewichtsscore in relatie tot elke andere patch, waardoor het model belangrijke gebieden binnen de afbeelding kan identificeren en zich daarop kan concentreren voor verdere verwerking. In het onderzoek van Bärudde et al. [11], is er verder ingegaan op het gebruik van vision transformer netwerken voor anomaliedetectie. Deze zouden state-of-the art performantie bieden ten opzichte van klassieke CNN's in een verscheidenheid aan vision tasks. Echter wordt een nadeel benadrukt: in tegenstelling tot een CNN komt een vision transformer doorgaans niet met een inductieve bias, waardoor er dus een grotere hoeveelheid data moet voorzien worden. Aangezien een nadeel van een CNN al de grote hoeveelheid data was, wordt een ViT netwerk in praktijk voor de industrie minder aantrekkelijk als er slechts een beperkt aantal samples aanwezig zijn. Ge et al. [12] hebben in hun onderzoek de toepasbaarheid van het Detection Transformer (DETR) netwerk onderzocht op het gebied van classificatie en segmentatie van vlakke houten platen. De resultaten

van hun studie wijzen uit dat DETR inderdaad geschikt is voor deze taken met een hoge performance aangegeven door een Mean Average Precision (MaP) van 95.5%. Deze score ligt 0.2% hoger dan die bereikt bij het gebruik van Mask R-CNN, wat wijst op een lichte maar significante verbetering in de nauwkeurigheid van objectdetectie. Het is echter belangrijk om te benadrukken dat de dataset gebruikt in dit onderzoek bestaat uit ongeveer 6000 afbeeldingen per foutcategorie, waarbij de geïdentificeerde fouten grootschalig zijn en gekenmerkt worden door duidelijk onderscheidbare kenmerken. Deze specifieke aard van de dataset kan een invloed hebben op de algemene toepasbaarheid van de resultaten. Met andere woorden, de effectiviteit van DETR in situaties met minder uitgesproken of kleinschaliger fouten blijft een vraag die verder onderzoek vereist.

2.2.2.3 Transfer learning

Zoals Ge et al. [12] al aangaven: een neuraal netwerk heeft een groot aantal afbeeldingen nodig om te trainen: bij voorkeur honderdduizenden. Dit is in praktijk niet haalbaar voor een bedrijf. Een oplossing hiervoor is het gebruik van reeds bestaande backbones. Dit zijn specifieke architecturen met een bewezen performantie waarvan de optimale gewichten gekend zijn doordat ze getraind zijn op een groot aantal afbeeldingen. Dit concept wordt transfer learning genoemd en biedt als significant voordeel dat de ontwerper niet meer op zoek moet naar een geschikte architectuur en bijhorende gewichten. Gao et al. [13] benadrukken wel dat transfer learning het training proces niet vervangt. De ontwerper dient nog steeds data aan te leveren, maar heeft wel de flexibiliteit om alleen de laatste laag te trainen of alle lagen fijn af te stemmen, waardoor het model nog nauwkeuriger kan worden.

Het onderzoek van Gao et al. [13] rechtvaardigt het gebruik van reeds bestaande classificatiemodellen vanwege de beperkte omvang van de beschikbare dataset. Het gebrek aan voldoende data zou anders de nauwkeurigheid van het uiteindelijke resultaat in gevaar kunnen brengen, aangezien het trainen met deze beperkte dataset zou kunnen leiden tot overfitting of underfitting. Het model dat door Goa et al. [13] wordt toegepast, beschikt al over een bestaande, getrainde image classification module. In hun onderzoek wordt een verhoogde nauwkeurigheid van 7,65% waargenomen, na het implementeren van transfer learning. Om een beeld te krijgen van de gebruikte pretrained backbones in de literatuur, werden verschillende image classification modellen vergeleken uit verschillende papers op basis van hun prestaties, samengevat in tabel 2.1. Hieruit kan besloten worden dat er geen algemene "one-size-fits-all" oplossing is. Echter heeft Resnet consistent een hoge performantie.

Tabel 2.1 Overzicht van prestatie-indicatoren van verschillende modellen.

Paper	Pretrained backbone	Precision	Recall	F1	Accuracy
[14]	MobileNetV3	97%	Nvt	Nvt	96%
[13]	TL-ResNet-34	98.32%	98.66%	98.46%	98.69%
[13]	AlexNet	94.77%	97.28%	95.91%	96.89%
[13]	VGGNet16	92.17%	96.86%	94.11%	95.58%
[13]	GoogLeNet	85.42%	91.62%	90.49%	95.42%
[1]	Resnet50	95.31%	Nvt	Nvt	Nvt
[15]	AlexNet	79.99%	80.01%	79.98%	Nvt
[15]	VGG16	78.04%	77.74%	77.16%	Nvt
[15]	BNInception	77.78%	78.06%	77.54%	Nvt
[15]	ResNet152	80.53%	80.65%	77.54%	Nvt

2.2.2.4 Hyperparameter tuning

Indien een geschikt model geselecteerd is, werd in vorig hoofdstuk aangehaald dat er nog steeds training nodig is. Echter is dit geen voorgedefiniëerd proces. Er zijn meerdere hyperparameters die invloed hebben op het training proces en tunebaar zijn. In verschillende onderzoeken werd snel duidelijk dat er geen universele parameters zijn die de meest optimale resultaten geven. Het tunen van deze parameters en het bijhorend resultaat na het trainen verschilt van case tot case. Hierdoor wordt in praktijk meestal een trial en error approach gebruikt. Volgende oplijsting bevat de meestvoorkomende hyperparameters uit de literatuur:

- **Batch Size**: Dit is het aantal trainingsexemplaren dat in één keer door het netwerk wordt verwerkt voordat de interne parameters worden bijgewerkt. Een grotere batch size kan de stabiliteit van de training verbeteren, maar vereist meer geheugen.
- **Learning Rate**: Dit is een maat voor hoe snel of langzaam het model leert. Een hogere learning rate kan leiden tot snellere convergentie, maar ook tot overslaan van de optimale oplossing. Een lagere learning rate maakt het leerproces stabieler, maar kan veel langer duren.
- **Epoch**: Dit is een term die een volledige doorloop van de trainingsdataset door het model aanduidt. Meer epochs betekenen dat het model meerdere keren de kans krijgt om van de gehele dataset te leren, maar ook dat het trainingstraject langer duurt. Een te lang trainingstraject kan echter voor overfitting zorgen.

- **Optimizer Functie**: Dit is het algoritme dat gebruikt wordt om de parameters van het netwerk aan te passen in de richting van de optimale oplossing. Verschillende algoritmes, zoals SGD, Adam, en RMSprop, hebben verschillende manieren van aanpassen, wat kan leiden tot snellere convergentie of betere prestaties op de trainingsdata.

2.2.3 Optimalisatie van hoge resolutie afbeeldingen

Zoals eerder besproken, vereist elk neuraal netwerk afbeeldingen als input. Hoewel sommige netwerken verschillende soorten afbeeldingen ondersteunen, hebben andere specifieke vereisten. In een studie door Benali Amjoud et al. [16] zijn veelgebruikte neurale netwerken en hun eigenschappen onderzocht en beschreven. Aan de hand van tabel 2.2 werd tot de conclusie gekomen dat de meeste neurale netwerken geen ondersteuning bieden voor hoge resolutie afbeeldingen.

Tabel 2.2 Inputgroottes voor verschillende neurale netwerken.

Neuraal netwerk	Input size
AlexNet	224x224
VGG-16	224x224
GoogLeNet	224x224
ResNet-50	224x224
ResNet-101	224x224
Inception-ResNet-V2	299x299
DarkNet-19	224x224

Zhang et al. [17] bespreken ook de input dimensies voor Yolo, omdat afbeeldingen eerst nog een segmentatie stap passeren. Wanneer Yolo een afbeelding met een grotere resolutie verwerkt, maakt het gebruik van een "herschaal" algoritme. Dit om aan de input voorwaarde van 416x416 pixels te voldoen. Stel dat een afbeelding van 1248x936 wordt ingevoerd, zal Yolo dit herschalen naar 416x312. Deze herschaling fungeert als een soort compressie, waardoor de resolutie van de afbeelding kleiner wordt en er ook details verloren gaan. Dit valt te verklaren aan de hand van het DORI-criteria. DORI staat voor: detection, observation, recognition and identification. De pixels van een object moeten minstens 10% zijn van de input om gedetecteerd te worden en 20% om gecategoriseerd te worden. Het onderzoek van Akyon et al. [8] gaat hierop verder en geeft aan de hand van de structuur van een convolutioneel neuraal netwerk, vier specifieke redenen waarom dit het geval is.

1. Een convolutioneel neuraal netwerk (CNN) maakt gebruik van filters, ook wel kernels genoemd. Deze filters reageren op specifieke delen van de invoer en hebben mogelijk

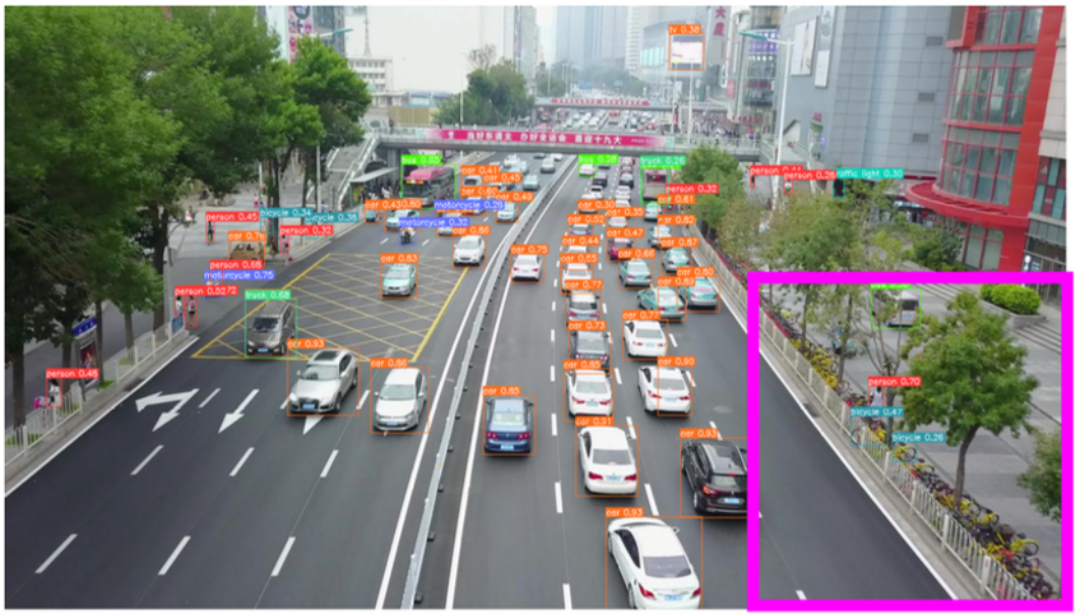
een beperkt receptief veld. Dit houdt in dat ze alleen informatie opvangen uit een beperkt gebied van de invoer. Als gevolg hiervan kan het gebruik van filters met een beperkt receptief veld leiden tot het verkleinen van de invoer, met als potentieel gevolg het verlies van details.

2. Bij het extraheren van kenmerken worden deze doorgaans doorgegeven aan een pooling layer. Deze pooling layer heeft de mogelijkheid om pixels samen te voegen, wat resulteert in het verlies van informatie met betrekking tot kleine objecten.

3. Bij transfer learning, zijn veelgebruikte pretrained modellen getrained op grote objecten. Hierdoor is er een bias en zal het model moeite hebben om kleine objecten te generaliseren.

4. CNN-netwerken kunnen mogelijk een beperkte ruimtelijke resolutie hebben, waardoor gedetailleerde informatie verloren kan gaan of onduidelijk kan worden ten opzichte van de achtergrond bij lage resolutie. Dit resulteert concreet in verwarring tussen kleine en grote objecten of de achtergrond.

Het onderzoek van Zhang et al [17] heeft tests uitgevoerd op praktijkgerichte afbeeldingen en kwam tot dezelfde conclusie dat neurale netwerken het moeilijk hebben om kleine objecten te detecteren. De wagens achteraan op figuur 2.2 werden niet gedetecteerd.

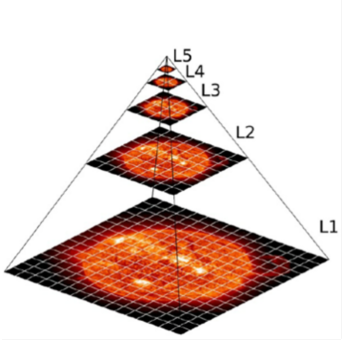


Figuur 2.7: Test neuraal netwerk op detailafbeelding [17]

De paper van Akyon et al. [8] bespreekt mogelijke oplossingen.

1. Image pyramids worden geïmplementeerd door te starten met het zoeken op het bovenste niveau, waar de resolutie het laagst is. Dit maakt het mogelijk om grove kenmerken

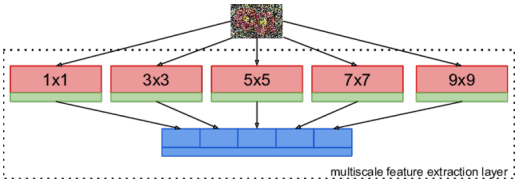
te identificeren. Vervolgens wordt op elk niveau van de piramide een objectdetectie-algoritme toegepast, waar kleinere objecten meer uitgesproken kunnen voorkomen en worden gedetecteerd. Ter illustratie figuur 2.8



Figuur 2.8: Image pyramids [17]

2. Het gebruik van sliding windows. De afbeelding wordt opgedeeld in windows, deze kunnen verschillende posities en schalen hebben. Het detectie algoritme werkt afzonderlijk op elk window, waardoor kleine objecten relatief groot worden ten opzichte van de window. Hierdoor wordt er voldaan aan het DORI-criteria.

3. Multiscale feature extraction. Het legt informatie vast op verschillende niveaus. Dit houdt in dat er meerdere convolutionele lagen gebruikt worden met verschillende receptive fields. Vervolgens is er interpolatie van verschillende features. Het nadeel hiervan is de verhoogde complexiteit van het neuraal netwerk wat als gevolg een verhoogde rekencomplexiteit met zich meebrengt. Dit komt de performance niet ten goede. In praktijk wordt SSD (Single shot multibox detector) hiervoor gebruikt. Ter illustratie figuur 2.9



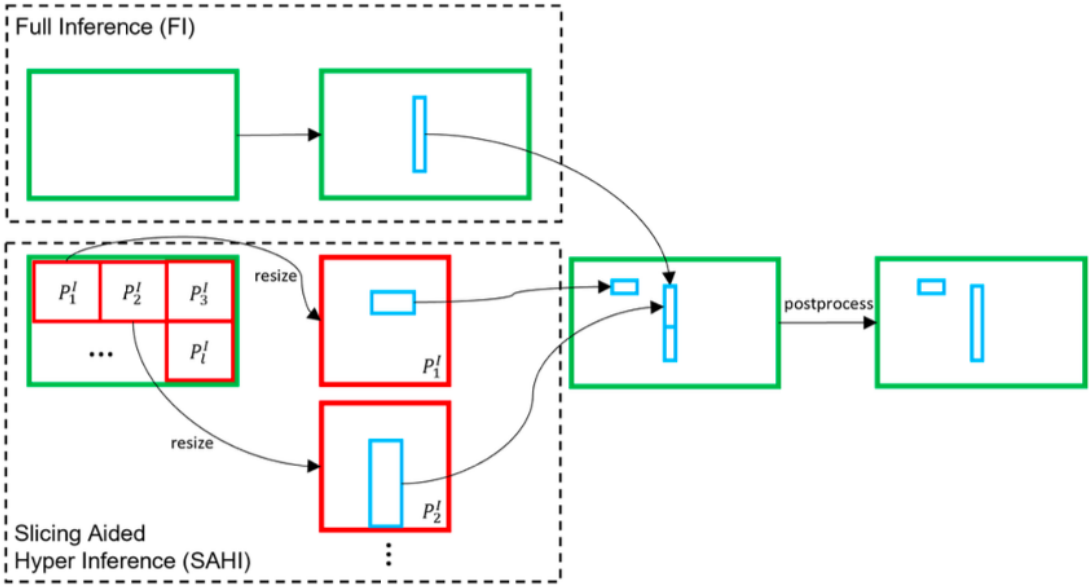
Figuur 2.9: Multiscale feature extraction [18]

4. Het toepassen van transfer learning met een bestaand model dat al is getraind op kleine objecten. Dit blijkt in de praktijk niet haalbaar te zijn, omdat er geen specifiek beschikbare open source CNN-modellen zijn die specifiek zijn getraind voor het detecteren van anomalieën in vlakke platen.

5. Data-augmentatie omvat het visueel aanpassen van afbeeldingen, zoals verticale uitrekking, om kleinere objecten beter zichtbaar te maken. Dit is een techniek die specifiek

geoptimaliseerd moet worden voor elke afzonderlijke afbeelding, waardoor het niet robuust is.

De paper van Akyon et al. [8] bouwt verder op de sliding window methode met Slicing Aided Hyper Inference (SAHI). Het is een framework dat verder bouwt op de sliding window techniek voor het detecteren van kleine objecten in beelden. De werking hiervan wordt duidelijk op figuur 2.10. Het werkt door het beeld op te splitsen in kleinere slices, waardoor de objectdetector (het neuraal netwerk) in staat is om meer details te zien en kleine objecten nauwkeuriger te detecteren. Concreet wordt de afbeelding opgedeeld in overlappende patches. De dimensies van deze patches kunnen gezien worden als hyper parameters. Ook is er een mogelijkheid om overlap en padding te creëren.



Figuur 2.10: Schematische weergave SAHI [8]

Nadat het neuraal netwerk elke patch heeft geanalyseerd, worden deze patches terug samengevoegd om zo een resultaat te bekomen. Deze techniek kan relatief eenvoudig in een pipeline geïntegreerd worden en vereist geen pretraining. Met deze methode is er een stijging van 14.5% in average precision (AP) voor kleine objecten en een overlap van 25% resulteert in een extra 2.9% AP. Echter wordt de computatie tijd lineair verhoogd.
Een ander domein waar dit probleem zich ook voordoet, is het analyseren van satellietafbeeldingen met computervisie. Hierover heeft het bedrijf genaamd "ML6" een blog geschreven. Gartz [19] schrijft over het gebruik van Slicing-Aided Hyper Interference om satelliet afbeeldingen te analyseren. ML6 gebruikt een dataset met relatief groote afbeeldingen van maximum 20.000 x 20.000 pixels. Ook het onderzoek van Gartz [19] rapporteerde hoge kwaliteitsresultaten voor de detectie van kleine objecten.

2.2.4 Imbalanced datasets

Imbalanced dataset doet zich voor wanneer één doelklasse een aanzienlijk deel van de waarnemingen vertegenwoordigt. Onevenwichtige datasets zijn datasets waarbij er een ernstige scheefheid is in de klassenverdeling, zoals bijvoorbeeld 1:100 of 1:1000 afbeeldingen in de minderheidsklasse ten opzichte van de meerderheidsklasse. Dit kan grote gevolgen hebben op de manier waarop het netwerk getraind wordt. Imbalanced datasets kunnen voor een permanente bias zorgen. Hierover schrijft Marcinrutecki [20] op het datawetenschapplatform genaamd Kaggle. Indien het netwerk getraind zou worden met als evaluatiefunctie de accuracy, zou er simpelweg een model kunnen ontstaan die alle afbeeldingen als foutloos categoriseert, en zo meer dan 99% accuracy haalt, omdat er minder dan 1% foutieve afbeeldingen zijn. Omdat dit niet wenselijk is, wordt er gekeken naar mogelijke oplossingen om een neuraal netwerk te trainen indien de dataset imbalanced is:

- Oversampling van de kleinste klasse, zo wordt de dataset terug gebalanceerd
- Undersampling van de grootste klasse
- Meer samples voorzien in de kleinste klasse
- Loss functie aanpassen

Meer samples voorzien zijn in praktijk meestal niet haalbaar. Ook het undersampelen van de grootste klasse is potentieel gevaarlijk. Dit wordt verklaard aan de hand van volgende analogie: indien een neuraal netwerk getraind wordt op het herkennen van handschrift, is het realistisch om evenveel afbeeldingen per letter te voorzien ter training. Dit werkt echter enkel in het geval van een hoge accuracy. Indien het neuraal netwerk onzeker is over een letter, is er nood aan een bias. Zo komt de letter “e” veel meer voor in woorden dan bijvoorbeeld de letter “z”. Als de training dataset dit reflecteert, kan deze bias ten goede komen voor het voorspellen van nieuwe afbeeldingen. Aan de andere kant is het gevaarlijk om een te hoge verhouding te hebben, bijvoorbeeld 1:1000.
Het onderzoek van Gao et al. [13] haalt de methode van oversampling van de kleinste klasse aan. Er worden meer samples gegeneerd door gebruik te maken van augmentatie. Augmentatie is het toepassen van filters en transformaties op afbeeldingen, om een afbeelding te bekomen die het neuraal netwerk waarneemt als “nieuw”. Ook zorgen deze filters en transformaties er mogelijks voor dat bepaalde features een verhoogde zichtbaarheid krijgen. In zijn onderzoek gebruikt hij volgende bewerkingen:

- Rotatie
- Verticale spiegeling
- Horizontale spiegeling
- Salt en pepper noise

- Gaussian noise
- Heu-saturation bewerkingen

Hierdoor is de dataset tot 7 keer groter geworden, wat het trainen van het neuraal netwerk ten goede komt. Een neuraal netwerk wordt accurater indien het getraind kan worden op meer samples. Dit omdat het dan de mogelijkheid ontwikkelt om beter te generaliseren en minder te overfitten. Een andere mogelijke techniek is het aanpassen van de loss functie. Standaard wordt cross-entropy het meest gebruikt als loss functie. Het meet de afwijking tussen de voorspelde kansverdeling en de werkelijke verdeling van de gegevens. Het bestraft onjuiste voorspellingen zwaarder en leidt het model effectief naar het minimaliseren van de ongelijkheid tussen zijn voorspellingen en de daadwerkelijke labels. Er is een variant beschikbaar voor binaire classificatie en categorische classificatie (indien er meerdere klassen zijn). Ook zou deze loss functie in staat zijn om goeie prestaties te leveren met imbalanced datasets. Echter zijn er ook andere loss functies die claimen robuust te zijn tegen imbalanced datasets:

- **Weighted cross entropy loss**: Er worden gewichten toegevoegd aan de klassen, op basis van hun frequentie, waardoor bepaalde klassen zwaarder doorwegen in het berekenen van de total loss.
- **Focal loss**: Ontwikkeld om het probleem van class inbalance aan te pakken. Het introduceert een modulerende factor die de bijdrage van gemakkelijk te classificeren voorbeelden vermindert.

Omdat accuracy geen realistisch beeld zou geven als performance metric, wordt het gebruik van recall, precision en een F-score evaluatiefunctie aanbevolen door Marcinrutecki [20].
Een eerste belangrijke term is recall:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.1}$$

Deze score geeft een goed beeld op het vlak van foutdetectie. Een hoge score is essentiëel waar het kritiek is om alle fouten te detecteren bijvoorbeeld in kankeronderzoek. Het nadeel van deze metric is dat het niets vertelt over de false positives en true negatives. Hiertegenover staat precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.2}$$

Deze vergelijking duidt aan hoeveel van de positieve gelabelde afbeeldingen effectief positief zijn.
Uit deze vergelijkingen is duidelijk dat recall en precision tegenstrijdige termen zijn. Indien het primeert om zoveel mogelijk fouten te detecteren (hogere recall), is de kans op false positives hoger (lagere precision). Verder praat Marcinrutecki [20] over formules die beiden uitbalanceren: indien de positieve klasse (de fouten) belangrijker zijn dan de negatieve klasse (achtergrond) wordt een F1 score aangeraden. Indien false positives minder erg zijn dan

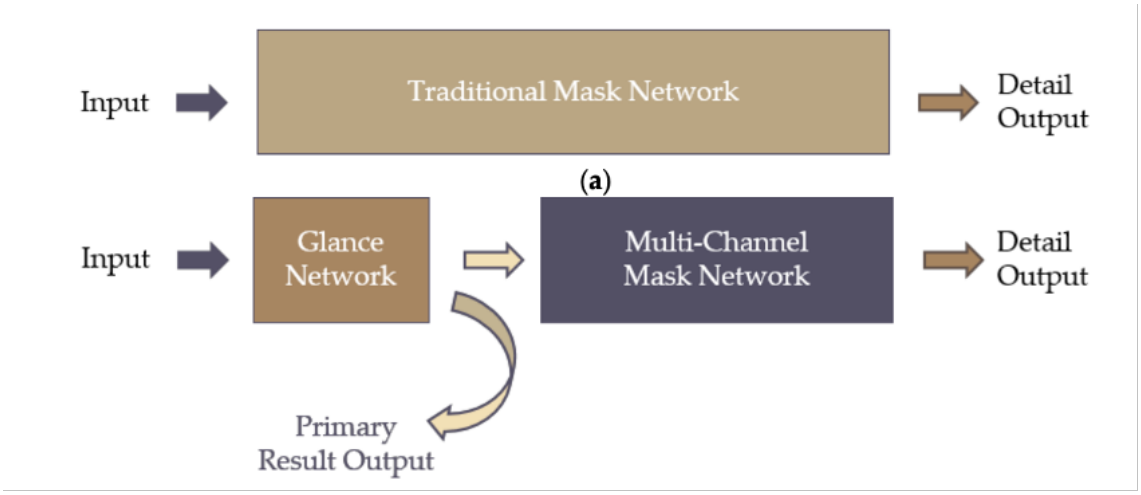
false negatives wordt de F2 Score aangeraden. Deze score vindt een optimale balans tussen precision en recall.

$$F1 \text{ score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.3}$$

$$F2 \text{ score} = \frac{5 \cdot \text{precision} \cdot \text{recall}}{4 \cdot \text{precision} + \text{recall}} \tag{2.4}$$

2.2.5 Interferentie optimalisatie

In dit hoofdstuk ligt de nadruk op de optimalisatie van de inferentietijd, oftewel de snelheid waarmee het algoritme functioneert. Gedurende het onderzoek van Girshick et al. [5] werd de trage inferentietijd als een van de grote nadelen van R-CNN beschouwd. Doordat het alle geselecteerde regio's moet verwerken, kan dit computationeel intensief zijn. Dit brengt het geschikt zijn voor realtime toepassingen in gedrang. Zo zou het minstens 9.8s nodig hebben voor interferentie op een enkele afbeelding van 416x416 pixels. Door het gebruik van Faster R-CNN, een geoptimaliseerde versie van R-CNN, wordt de inferentietijd met factor 80 versneld. Indien dit gecombineerd wordt met een SAHI-methode, duurt het 12.3s om 100 afbeeldingen te verwerken. Voor bepaalde productie toepassingen kan dit te traag zijn. Het onderzoek van Shi et al. [1] introduceert een innovatieve oplossing dat twee netwerken combineert. De input wordt eerst verwerkt door een glance netwerk. Dit is een CNN met een eenvoudige architectuur dat als doel heeft een window binair te classificeren. Pas indien er een window fout is, wordt dit doorgegeven aan een meer complexe classificatie en segmentatie netwerk, bijvoorbeeld mask-RCNN. De achterliggende filosofie van dit algoritme is gebaseerd op het idee dat het verwerken van alle vensters door het mask-RCNN netwerk te tijdrovend zou zijn en te veel rekenresources zou verbruiken. Het glance network biedt aanzienlijk snellere verwerkingssnelheden. Ter illustratie figuur 2.11.



Figuur 2.11: Vergelijking traditioneel mask netwerk en Glance netwerk [1]

Shi et al. [1] duiden dat reeds bestaande backbones toch nog computationeel complex kunnen

zijn. Daarom stellen ze hun eigen architectuur op, gebruik makende van NAS-technologie, wat staat voor Netwerk Architectuur Search. Deze aanpak stelt automatisch de meest optimale netwerkarchitectuur voor, met in dit onderzoek een specifieke focus op snelheid, een cruciale vereiste voor het glance-netwerk. Na 66 uur zoeken is uiteindelijk een architectuur ontdekt, weergegeven in figuur 2.12, bestaande uit een convolutionele laag met 280 trainbare parameters, een max-pooling laag, een flatten laag en een dense laag met 800000 trainbare parameters. Deze architectuur is beduidend eenvoudiger dan bekende bestaande backbones, wat resulteert in een lagere inferentietijd.

Figuur 2.12: Glance netwerk architectuur [1]

Vervolgens wordt het effect van een "2 model approach" bekeken op de performantie metrics van de test dataset. Er is aanzienlijke vooruitgang geboekt op het gebied van MAP, met een indrukwekkende score van 95,31%. Het meest opvallende verschil is echter te zien in de inferentietijd voor een batch. Met een verwerkingstijd van 2.5 seconden voor een batch van 100 afbeeldingen van 200x200 pixels, steekt dit algoritme boven de competitie uit qua performantie. De samenvatting van de vergelijking is terug te vinden in figuur 2.13.

Method	OCA (%) [1]	MAP (%) [2]	Inference Time/Batch (s)
GM-Mask R-CNN (Resnet50) [3]	98.70	95.31 ± 4.5	2.5
Mask R-CNN (Resnet101)	98.52	93.32 ± 3.2	6.1
SegNet [48]	98.45	92.27 ± 3.9	20.1
FCN [4] [49]	98.45	90.67 ± 4.1	9.7

[1] OCA is overall classification accuracy. [2] MAP is mean average precision. [3] GM-Mask R-CNN is glance network and multiple channels mask region convolution neural network. [4] FCN is fully convolutional neural network.

Figuur 2.13: Vergelijking GM-Mask R-CNN met andere architecturen [1]

De inferentietijd blijft echter een variabel gegeven, afhankelijk van de specifieke testopstelling en de gebruikte hardware. In deze paper werd gebruikgemaakt van een NVIDIA GeForce RTX 2080 TI in combinatie met 32 GB RAM en een Intel Core i7-8700. Hoewel deze hardware krachtiger is dan de gemiddelde consumentencomputer, blijft het een haalbare investering voor bedrijven die dergelijke concepten willen implementeren.

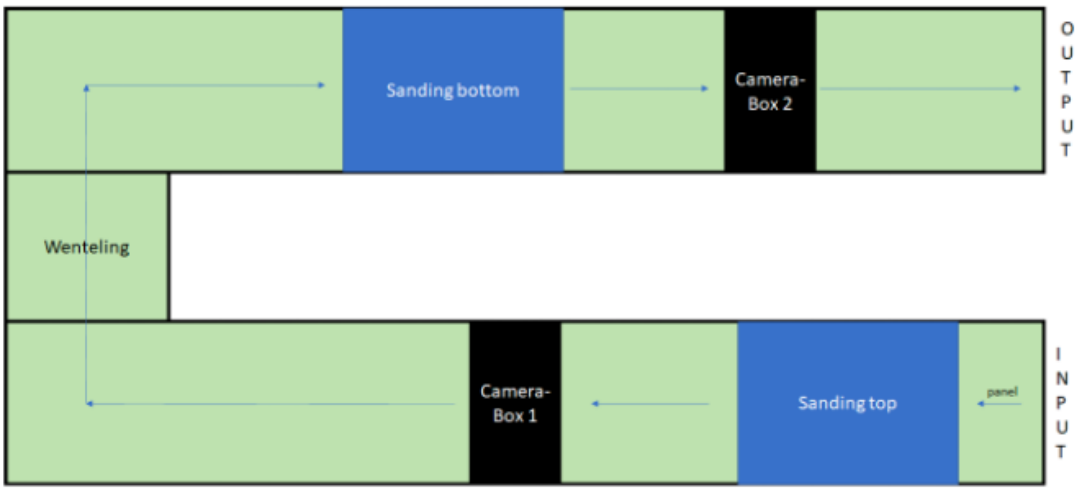
Hoofdstuk 3

Methodologie

In dit hoofdstuk wordt de literatuurstudie tegenover het praktische gezet. Concreet wordt er gekeken naar de methodologieën om onderzochte aspecten van deep learning en machinevisie praktisch te implementeren en dichter bij een antwoord op de onderzoeksvragen te komen.

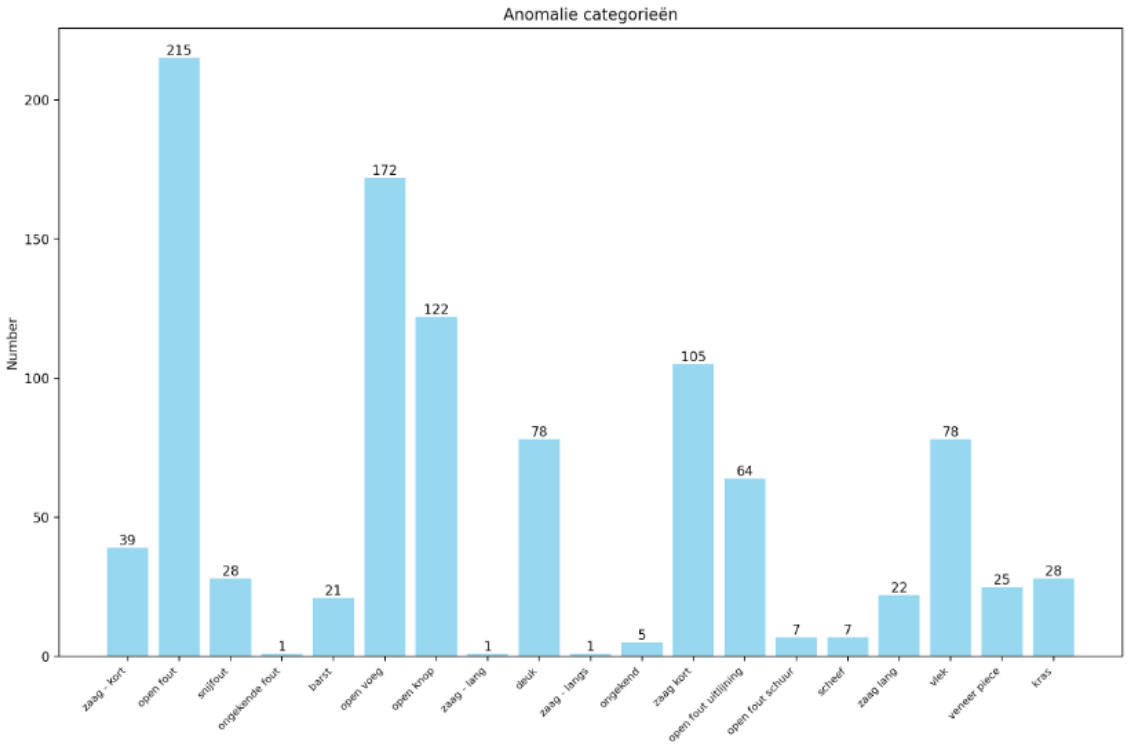
3.1 Meetopstelling en dataset

Figuur 3.1 bevat een overzicht van de laatste stap van het productieproces van Decospan. Tijdens deze stap worden fineerplaten aan beide kanten geschuurd. Zoals te zien op de figuur zijn er twee lijnscancamera's geïmplementeerd om elk een kant van de fineerplaat te fotograferen. Dit proces is volledig geautomatiseerd en start met het detecteren van de groene band (achtergrond) als start en stopsignaal. Vervolgens legt de lijnscancamera meerdere afbeeldingen vast die softwarematig aan elkaar gemonteerd worden. Om hier een correcte opname te garanderen, wordt er gebruik gemaakt van een encoder die inschat hoeveel centimeter het paneel verschoven is op de productielijn.

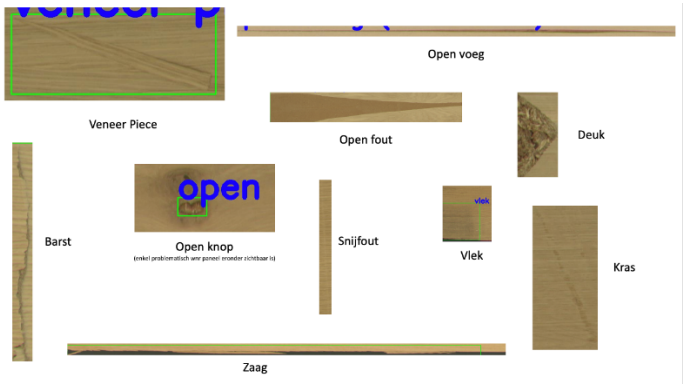


Figuur 3.1: Productielijn Decospan

Door de meetopstelling is het mogelijk een globaal beeld te krijgen van elke fineerplaat. Om een grondige analyse te kunnen voeren van de voorkomende fouten en zo een dataset op te stellen, duidt een medewerker van Decospan elke anomalie manueel aan met de annotatiesoftware "LabelImg". Dit resulteert in 371 afbeeldingen, die mogelijks een of meerdere fouten bevatten. In figuur 3.2 worden het aantal fouten en de bijhorende foutcategorieën weergegeven. Figuur 3.3 geeft een visuele weergave van de meest voorkomende foutencategorieën.



Figuur 3.2: Oorspronkelijke dataset Decospan

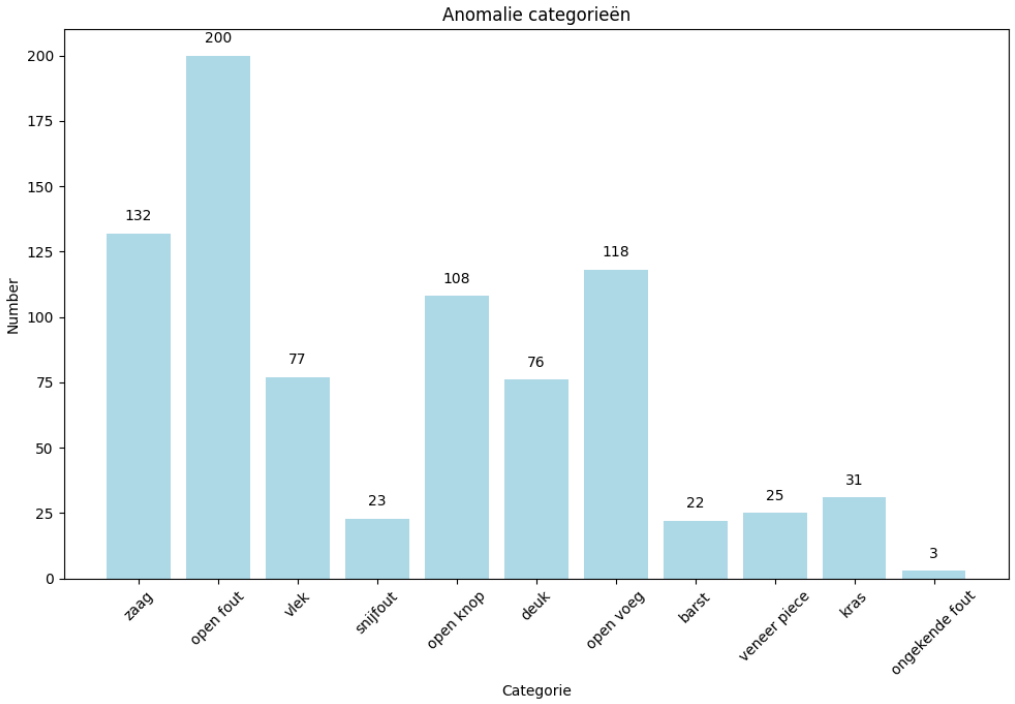


Figuur 3.3: Visuele weergave veelvoorkomende fouten

Uit de figuur wordt duidelijk dat er meerdere foutcategorieën zijn. Om te zorgen dat het neuraal netwerk een minder complexe architectuur heeft, dienen het aantal categorieën zoveel als mogelijk beperkt te worden. Daarom duidt tabel 3.1 aan welke categorieën samengevoegd worden. Ook werd de categorie scheef verwijderd, omdat de fineerplaten standaard niet perfect loodrecht gefotografeerd zijn. Hierdoor valt er niet te onderscheiden of de plaat "scheef" is of de plaat niet-loodrecht gefotografeerd is. Het resultaat hiervan is te zien op figuur 3.4.

Tabel 3.1 Aangepaste foutcategorieën

Samengevoegde categorieën	Verklaring
Zaag – lang, zaag – kort	Duidt aan of de fout zich in de lengte of breedte bevindt. Dus kan nadien bekeken worden aan de hand van coördinaten.
Open fout, open fout uitlijning, open fout schuur	Visueel zelfde fout, duidt ook aan waar de fout zich bevindt.

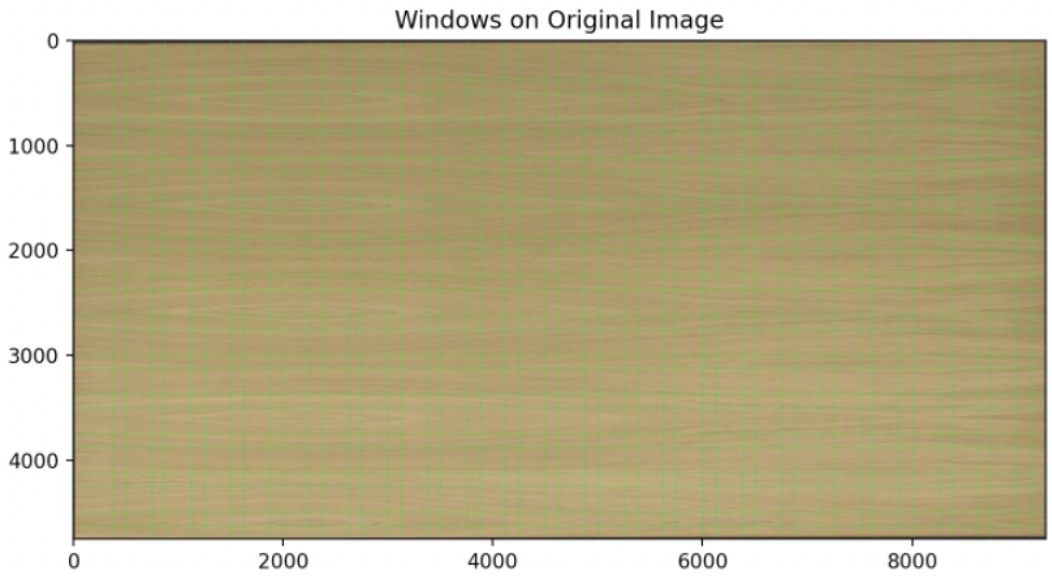


Figuur 3.4: Aangepaste dataset Decospan

3.2 SAHI: verwerking van hoge resolutie afbeeldingen

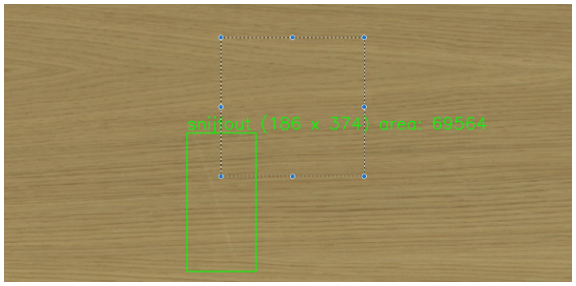
Het algoritme start met het verwerken van grote resolutie afbeeldingen aan de ingang. Hierdoor is er nood aan een strategie om hoge resolutie afbeeldingen te kunnen verwerken en vervolgens te "voeden" aan het neurale netwerk. In hoofdstuk 2.2.3 werden meerdere mogelijkheden voorgesteld. In het geval van de Decospan dataset blijft de aard van de fout echter de bottleneck voor veel algoritmes: de fouten zijn zeer klein en moeilijk met het blote oog te zien. Hierdoor is het noodzakelijk om elke pixel van de plaat afzonderlijk te analyseren en bieden technieken als een gelaagde analyse op basis van een piramide geen efficiëntievoordelen buiten een performantiewinst op vlak van inferentietijd. Het onderzoek van Akyon et al. [8] bevestigde ook al dat SAHI geschikt is voor het analyseren van detailfouten. Concreet wordt

de hoge resolutie afbeelding in deze stap opgedeeld in vierkante windows van 448x488 pixels met een overlap van 25%, zoals weergegeven in figuur 3.5. Deze keuze is gebaseerd op de input grootte van neurale netwerken en het DORI-criteria beschreven in punt 2.2.3. Door 2x de input grootte te kiezen van het neuraal netwerk is er slechts 50% compressie nodig, waardoor het gedetailleerde karakter van de fouten duidelijk blijft en er geen overvloedig aantal windows zijn, waardoor de inferentietijd per plaat te hoog zou worden.



Figuur 3.5: SAHI-algoritme

Aangezien supervised learning manuele annotaties verwacht en unsupervised manuele annotaties nodig heeft om een performantiescore te leveren, doet zich een nieuwe uitdaging voor: de huidige SAHI-methode is niet compatibel met de bounding box annotatie van Decospan. De bounding boxes geleverd door Decospan bevatten weliswaar de fout, maar zijn te ruim genomen. Door het SAHI-algoritme wordt deze bounding box dan verdeeld in stukken, die allemaal als foutief gelabeld worden, terwijl dit niet voor alle windows het geval is. Figuur 3.6 geeft hiervan een voorbeeld. Deze bounding box (aangegeven met een groene volle lijn) bevat effectief een snijfout. Vervolgens wordt een voorbeeld SAHI window weergegeven (stippellijn). Aangezien er een overlap is tussen beiden zou dit window als "foutief" gelabeld worden, terwijl het de fout niet bevat en zo een neuraal netwerk kan "verwarren" tijdens training.



Figuur 3.6: SAHI-algoritme in combinatie met bounding boxes

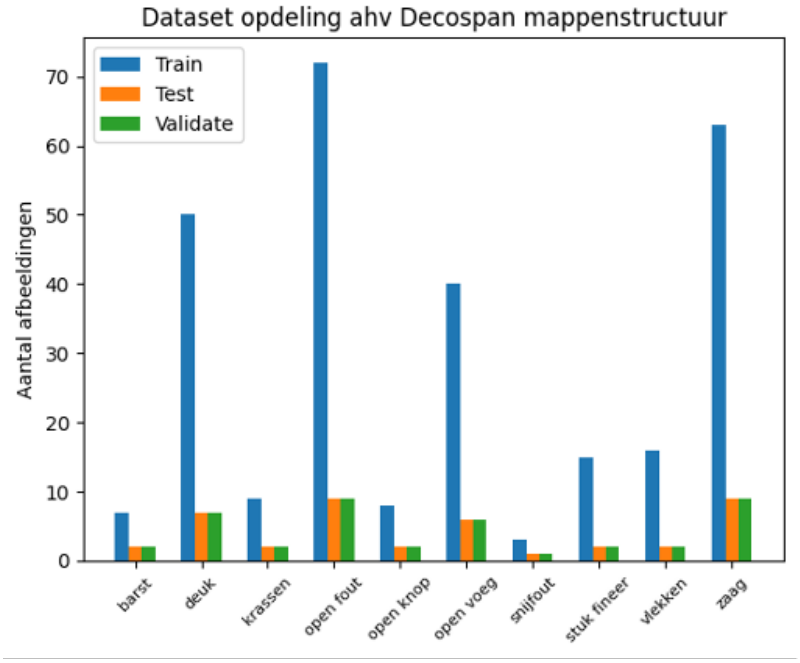
Een oplossing hiervoor is het herannoteren van de dataset. Er wordt overgestapt op een polygonaal annotatie. Hierdoor bevat de geselecteerde foutenzone effectief de fout en zal dit dus geen probleem meer geven met het SAHI-algoritme. Hierdoor staat de dataset ook al in het juiste formaat om het Mask R-CNN netwerk te trainen. Een voorbeeld annotatie is te zien in figuur 3.7. De software genaamd "LabelMe" [21] werd gebruikt voor de polygonale annotaties.

Figuur 3.7: Polygonale annotatie

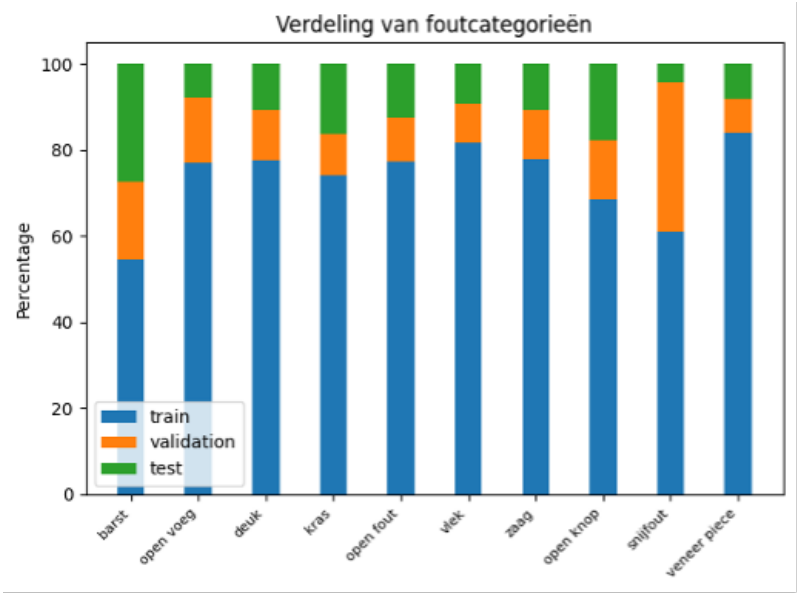
3.3 Imbalanced dataset

Zoals de literatuur het voorschrijft wordt de data opgesplitst in drie categorieën: train, validate en test. Hier is het van belang om "data leakage" te vermijden. Training gebeurt strikt op de train dataset. Tijdens het trainen is er validatie met de validate dataset, om zo een beeld te krijgen van de kwaliteit gedurende het trainproces en de hyperparameters fine te tunen. Nadien wordt de accuracy, recall en precision van het bekomen model getest met de test dataset. Hiervoor dient een ratio gekozen te worden. Er wordt gekozen voor 80-10-10, om de training dataset te maximaliseren, wegens het beperkt aantal afbeeldingen die Decospan aanlevert. Om de verdeling effectief uit te voeren dient er een methode gekozen te worden. Voor de hand liggend is het "ad random" verdelen van de background en faulty dataset volgens de verhouding. Echter is dit in het geval van Decospan niet mogelijk omwille van de herkomst van de windows: sommige windows komen van dezelfde bronafbeelding en

kunnen dus niet gesplitst worden over deze drie categorieën. Indien er getest zou worden met bijvoorbeeld een background, afkomstig van dezelfde bronafbeelding waar het model op getraind is, zou overfitting niet opvallen en zou het model een schijnbare hoge accuraatheid hebben. Verder is het ook belangrijk om de type fouten evenredig te verdelen over deze drie subsets. Zo kan er gegarandeerd worden dat het model getraind is op elke soort fout. Hiervoor wordt gebruik gemaakt van stratified split methode. Dit is een algoritme die ernaar streeft elke klasse evenveel te vertegenwoordigen in de verschillende opsplitsingen van de dataset. In figuur 3.8 en 3.9 wordt het resultaat van deze splitsing weergegeven. De y-as representeert het aantal instanties van elke fout. Echter is het door de opdeling van de fouten per afbeelding niet mogelijk om een perfecte 80-10-10 split over de train-test-validate te doen, zoals de literatuur het aanraadt.

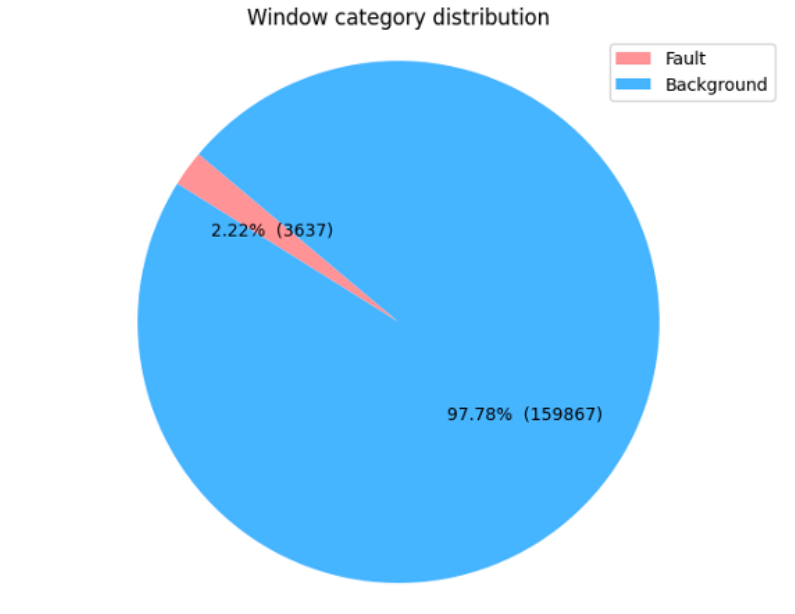


Figuur 3.8: Instanties per fout in aantallen



Figuur 3.9: Instanties per fout in aantallen in pct

Na het verdelen en analyseren van de dataset van Decospan wordt duidelijk dat het grootste oppervlakte van de fineerplaten foutloos is en hierdoor in de "background" categorie valt. Dit zorgt ervoor dat de dataset niet in balans is. Figuur 3.10 drukt, op basis van gegenereerde windows afkomstig van het SAHI-algoritme, de verdeling uit.



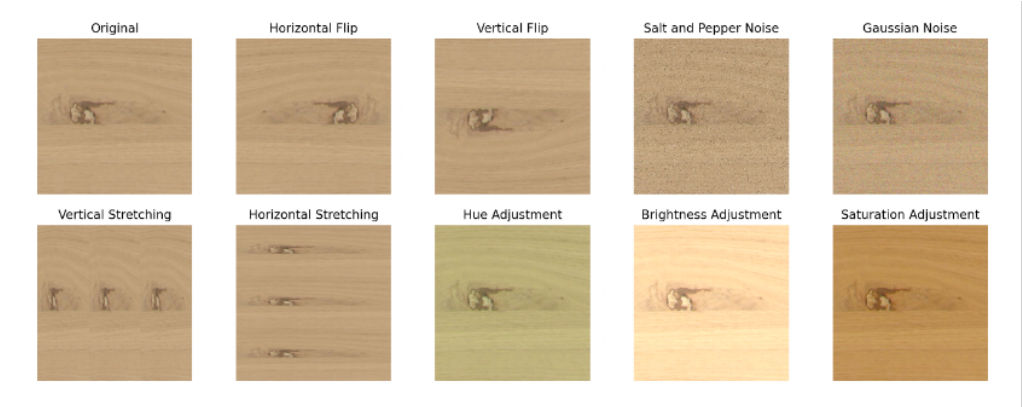
Figuur 3.10: Aantal windows binair gezien

Aangezien slechts 2.22% van de windows een fout bevat, zou het trainen van een neuraal netwerk met deze dataset, zoals besproken in hoofdstuk 2.2.4, potentieel gevaarlijk zijn en

voor een verlaagde accuracy zorgen. Om dit tegen te gaan, wordt er gekeken naar 2 mogelijke manieren: het sampelen van de data en het voorzien van meer samples aan de hand van data augmentatie.

Dataset sampling: Het machine learning framework "PyTorch" heeft een ingebouwde functie genaamd: "WeightedRandomSampler". Deze functie heeft als doel het oplossen van de inbalans in een dataset. Door elke klasse een gewicht toe te kennen, houdt Pytorch rekening met de frequenties per klasse. Vervolgens kiest Pytorch een gebalanceerd aantal afbeeldingen uit de dataset per batch. Dit heeft als voordeel dat elke batch andere afbeeldingen heeft. Door geen statische selectie over alle batches te nemen, is de kans op verlies van belangrijke data van de hoogfrequente klasse kleiner.

Dataset augmentatie: Om deze inbalans te verbeteren kan er ook gebruik gemaakt worden van augmentatie technieken. Er werden 9 augmentaties toegepast per window van de foutcategorie. Dit resulteerde in een verhoging van 10.000% in windows die fouten bevatten. Figuur 3.11 visualiseert de augmentaties.



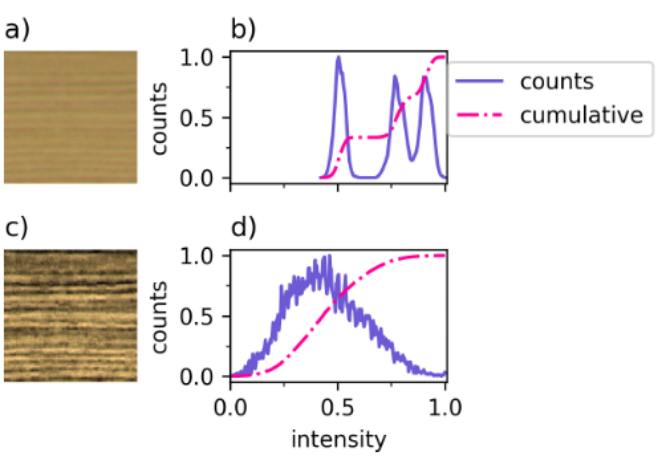
Figuur 3.11: Augmentaties

3.4 Unsupervised learning

3.4.1 K-means clustering

Voor het K-means clustering algoritme wordt de dataset gesampled. Elke test bevat standaard alle foutieve windows, gedefinieerd in hoofdstuk 3.3. Dit om de clustervorming zo accuraat mogelijk te maken en robuuster te zijn tegen outliers. Afhankelijk van de type test kunnen random gesamplede background windows bijgevoegd worden. Hierbij is er een balans tussen het representeren van genoeg houtsoorten, zonder teveel computationele complexiteit toe te voegen door het verwerken van een grootschalig aantal windows. De performantie wordt in kaart gebracht door testen uit te voeren met binaire en multi-class clustering. Hierbij wordt het effect van hyperparameter tuning besproken voor parameters als dimensionaliteits-

reductie, backbone en laagselectie. Zoals beschreven bestaat de K-means clustering pipeline uit drie delen. Een eerste deel is het preprocessen van de windows. Elke window ondergaat een resizing naar 224x224 pixels, omdat de VGG16 backbone dit verwacht. Vervolgens volgt een histogram equalisation, zoals Cohn et al. [2] aanraden. Op figuur 3.12 is het effect van histogram equalisatie te zien, het contrast wordt verhoogd, zodanig het normaal verdeeld wordt.



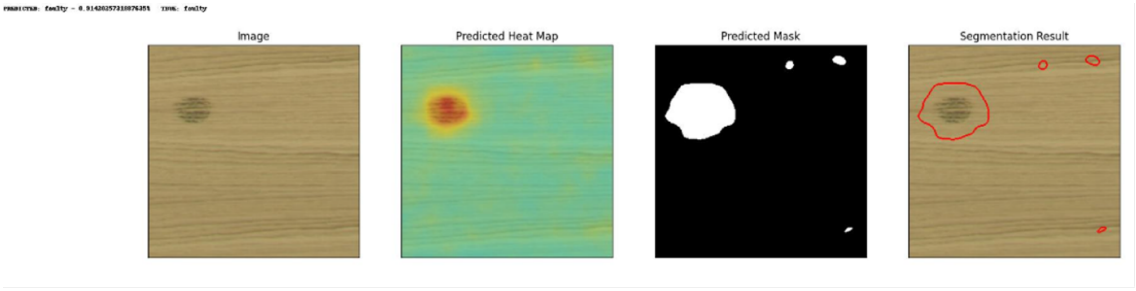
Figuur 3.12: Histogram equalisation

Een tweede stap is het extraheren van features. Omdat dit een trial en error proces is worden meerdere lagen getest op de efficiëntie. Van vgg16 worden er testen uitgevoerd op de eerste, tweede en derde fully connected layer. Als de features geëxtraheerd zijn wordt er PCA uitgevoerd omwille van het grootschalig aantal parameters. Het doel van PCA is de dimensionaliteit verminderen en het wegfilteren van ruis. Ook hier is het aantal dimensies een proces van trial en error. Cohn et al. [2] raden 50 dimensies aan. Tenslotte volgt clustering. Hier dient het aantal trials en het aantal clusters gekozen te worden. In iedere trial zal het algoritme een andere seed nemen om centroïden van clusters te genereren. Cohn et al. [2] gebruiken zeven clusters, analoog aan het aantal klassen en 500 trials.

3.4.2 PaDiM

Voor anomaliedetectie met PaDiM wordt er gewerkt met de Anomalib [22]. Het biedt verschillende kant-en-klare implementaties van algoritmen aan voor anomaliedetectie, evenals een reeks hulpmiddelen die de ontwikkeling en implementatie van aangepaste modellen faciliteren. Het is onderdeel van de open OpenVino toolkit van Intel. Standaard is deze geoptimaliseerd voor intel CPU's, waardoor conversie vereist is naar andere architecturen zoals Nvidea of Amd. Voor deze testen werden enkel de background windows, gedefinieerd in hoofdstuk 3.3, gebruikt ter training. Ook vinden er specifieke testen plaats: het trainen van een model op alle background windows, een subsample, een subsample zonder randen en een subsample per houtsoort. De sampling gebeurt steeds random met windows uit de originele dataset.

De Anomalib implementeert een pipeline: als preprocessing stap wordt er genormaliseerd en is er een mogelijkheid om augmentaties in combinatie met de albumentations library toe te passen. Vervolgens is het mogelijk hyperparameters in te stellen voor de training zoals batch size, number of workers, etc. Tenslotte is er een ingebouwde visuele weergave, zichtbaar in figuur 3.13, om inzicht te krijgen in de beslissingsstructuur van het algoritme. Dit biedt een voordeel in kader van de "explainability" van AI.



Figuur 3.13: PaDiM example

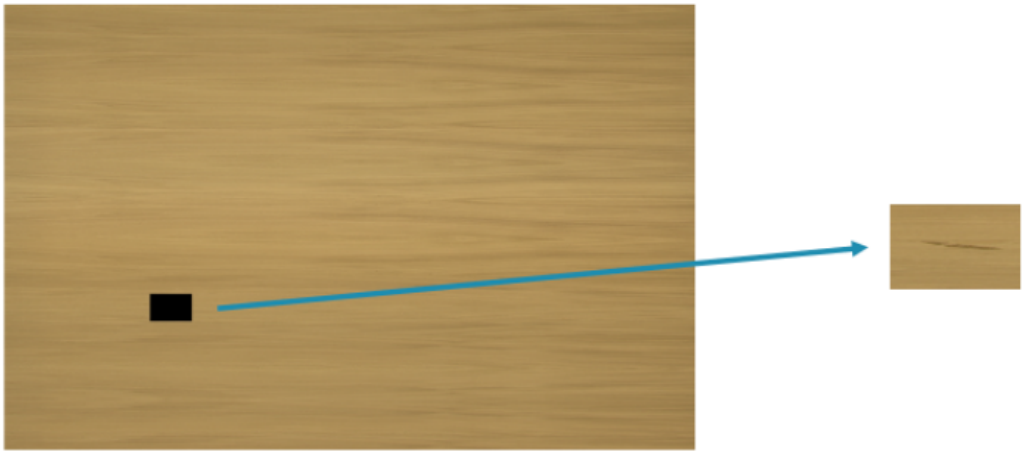
3.5 Supervised learning

De onderzoeksvragen in het achterhoofd houdende, is er een vergelijking nodig tussen het binair en multi-class classificeren van de Decospan dataset. Hiervoor kan er teruggekoppeld worden naar de literatuurstudie: daar worden zowel classificatie- als segmentatie modellen aangehaald. Tegelijk wordt de transfer learning techniek toegepast, omdat de dataset van Decospan beperkt is.

3.5.1 Binaire CNN

Voor dit deel worden de labels van de alreeds gesplitste Decospan dataset gereduceerd naar twee categorieën: "fault" en "background. Uit een literatuurstudie zijn twee "state-of-the-art" modellen geïdentificeerd die veelbelovend lijken voor de taak van binaire classificatie: YOLO en ResNet. Bij beide modellen wordt de laatste laag aangepast, met als doel slechts twee klassen te voorspellen. Voor de implementatie van het YOLO-model wordt er gekozen voor de 's'-versie, die minder rekenkracht vereist en daardoor compatibel is met de meeste computersystemen. Wat betreft het ResNet-model, zullen varianten met 18, 34 en 50 lagen worden toegepast om het effect van een toenemend aantal lagen op de prestaties en complexiteit te onderzoeken. Hoewel een hoger aantal lagen kan leiden tot een verhoogd risico op overfitting, bestaat de mogelijkheid dat deze modellen beter presteren op complexe datasets. Ter vergelijking wordt ook een specifiek model ontwikkeld voor elke houtsoort in de dataset. Deze aanpak beoogt de prestaties van de modellen te maximaliseren door ze af te stemmen op de unieke kenmerken van elke houtsoort. Tenslotte wordt teruggekeken op de bevindingen uit hoofdstuk 2.2.3, waarin werd geconstateerd dat de overlapping van manuele

annotaties en SAHI-windows niet volledig foutloos zijn. Deze onnauwkeurigheden kunnen ertoe leiden dat het SAHI-algoritme vensters ten onrechte als foutief classificeert (false positives). Bovendien bestaat de kans dat het netwerk traint op vensters waarin een fout slechts gedeeltelijk aanwezig is. Als alternatief hiervoor is een "cutout"-methode ontwikkeld, een eigen ontworpen methode. Deze methode knipt de fouten vooraf uit de afbeelding (met een passende marge), om vervolgens zwarte padding achter te laten. Dit resulteert in een dataset waarin de fouten expliciet en centraal binnen een venster worden weergegeven, waardoor het model eenduidiger kan trainen. Ter illusratie figuur 3.14. Vervolgens wordt de performantie van deze methode vergeleken met de klassieke SAHI-methode.



Figuur 3.14: Cutout-method example

3.5.2 Multi-class CNN

Dit convolutionele neurale netwerk (CNN) is specifiek ontworpen voor het classificeren van afbeeldingen in een van de twaalf categorieën, bestaande uit elf foutcategorieën en één achtergrondcategorie. Voor de ontwikkeling van dit model wordt voortgebouwd op de architectuur van de eerder gebruikte binaire CNN-modellen, namelijk YOLO en ResNet. Echter, voor deze toepassing is de outputlaag van de modellen aangepast om twaalf klassen te kunnen onderscheiden in plaats van twee. Ook hier is er aandacht voor de onderzoeksvragen: er is een directe vergelijking tussen training op enkel fouten en training op fouten en backgrounds. Hiervoor wordt de dataset, gedefinieerd in hoofdstuk 3.3 gebruikt. Voor de eerste test worden alle foutieve windows gebruikt. Daaropvolgend, voor een tweede test, worden een gelijk aantal gesamplede background windows toegevoegd, ad random gekozen, om elke houtsoort zo representatief mogelijk in te calculeren in de training.

Hoofdstuk 4

Implementatie

4.1 Dataset

Om resultaten te verkrijgen die reproduceerbaar en vergelijkbaar zijn doorheen verschillende testen is steeds eenzelfde dataset gebruikt. Hiervoor werd de splitsing weergegeven in figuur 3.9 en 3.10 gebruikt. Dit resulteert in 3637 foutieve windows en 159.000 background windows, weergegeven in figuur 3.11. Tabel 4.1 bevat het aantal windows per dataset split per foutcategorie. Hieruit wordt gesubsampled, afhankelijk van het type test. Voor een training op bijvoorbeeld enkel foutieve windows, worden de foutieve windows buiten beschouwing gelaten. De verdeling train-validate-split blijft steeds behouden en consistent over testen heen. Performantie metrics, besproken in tabellen van hoofdstuk 4, zijn steeds toegepast op de test dataset.

Tabel 4.1 Verdeling van data over trainings-, validatie- en testsets per categorie.

Categorie	Train	Validate	Test
Open fout	933	139	137
Open voeg	520	172	46
Deuk	128	29	44
Zaag	910	149	132
Barst	103	18	18
Open knop	147	28	40
Snijfout	45	13	9
Veneer piece	74	21	29
Vlek	242	26	28
Ongekend	4	2	0
Kras	167	72	34
Background	114402	22206	23259

4.2 Unsupervised learning

Deze testen proberen een antwoord te formuleren op de onderzoeksvraag: "Kan een (unsupervised) anomaliedetectie model gebruikt worden om de defecten af te zonderen?". Eerst wordt gestart met K-means clustering, vervolgens anomaliedetectie met PaDiM.

4.2.1 K-means clustering

Zoals beschreven in de methodologie wordt er onderscheid gemaakt tussen de verschillende klassen door middel van clusters te vormen. Ter illustratie figuur 4.1.

Figuur 4.1: K-means clustering example

Voor deze testen kunnen 4 soorten hyperparameters ingesteld worden:

1. Selectie backbone
2. Laagselectie van backbone
3. Multi-class of binair
4. Dimensionaliteitsreductie

Selectie backbone: Cohn et al. [2] beschreven de keuze van een backbone als een trial en error proces. Daarom werd er een vergelijkende test uitgevoerd tussen twee bekende backones: vgg16 en resnet50. VGG16 is ontworpen met een zeer uniforme structuur, bestaande uit opeenvolgende convolutielagen gevolgd door max-pooling, wat resulteert in een relatief eenvoudig en diep netwerk. Aan de andere kant implementeert ResNet-50 een dieper, maar meer geavanceerd netwerk dat gebruik maakt van "residual learning" blokken. Deze residual blokken helpen het netwerk om kenmerken op verschillende abstractieniveaus efficiënter te leren. Tabel 4.2 toont het effect van een backbone op de accuracy. Hiervoor zijn enkel de 3273 foutieve windows uit de dataset van 4.1 gebruikt, zodat er 11 klassen mogelijk zijn, analoog aan figuur 2.1. Hieruit blijkt dat de geavanceerdere structuur in combinatie met de residual learning blokken een hogere performantie geven op de Decospan dataset.

Tabel 4.2 Nauwkeurigheid van verschillende backbones.

Backbone	Layer	Accuracy
Vgg 16	FC 1	0.452
Resnet 50	FC 1	0.521

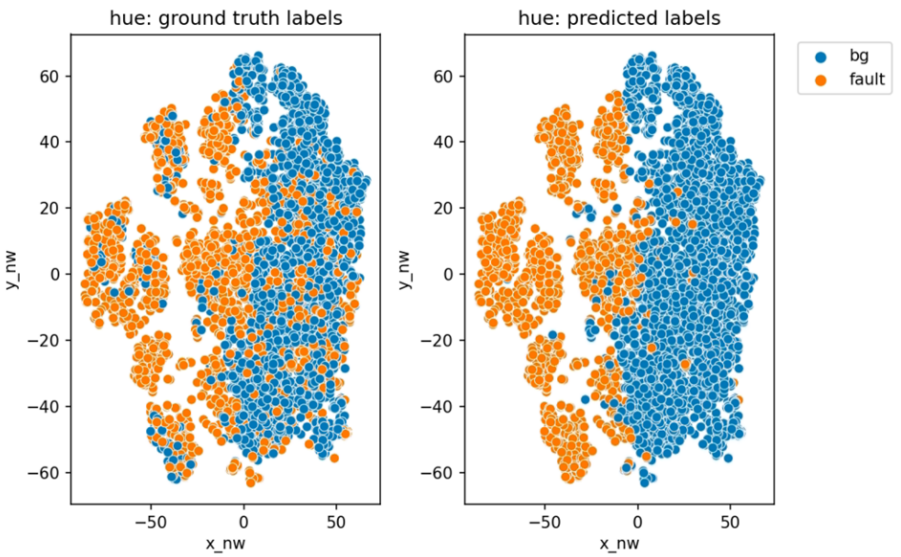
Laagselectie van backbone: Zoals beschreven in de methodologie kan de geselecteerde laag een effect hebben. Hiervoor werden testen uitgevoerd op vgg16, aangezien deze drie fully connected layers heeft. Resnet 50 heeft er slechts een. De dataset blijft hetzelfde als tabel 4.2. Tabel 4.3 geeft het resultaat hiervan weer.

Tabel 4.3 Nauwkeurigheid van verschillende lagen binnen gekozen backbones.

Backbone	Layer	Accuracy
Vgg 16	FC 1	0.452
Vgg 16	FC 2	0.410
Vgg 16	FC 3	0.423
Resnet 50	FC 1	0.521

Hieruit blijkt dat de keuze van een laag ook een trial en error proces is. Er zijn verschillende redenen waarom het kiezen van een andere laag niet noodzakelijk een concrete verbetering biedt voor het extraheren van features: netwerken met verschillende diepten en configuraties ontwikkelen verschillende representaties op hun lagen, wat betekent dat een laag in het ene netwerk niet noodzakelijkerwijs vergelijkbaar is met dezelfde laag in een ander netwerk. Er is een aanzienlijke variabiliteit in wat verschillende lagen leren, zelfs binnen hetzelfde netwerk.

Binaire classificatie: In plaats van enkel de foutieve windows van de dataset te gebruiken, kunnen background windows toegevoegd worden. Om de computationele complexiteit te beperken, is er op random basis een evenredig aantal (3273) background windows geselecteerd, om zo de verschillende houtsoorten te vertegenwoordigen. De 11 foutcategorieën worden dan samengevoegd naar een categorie: "fault".



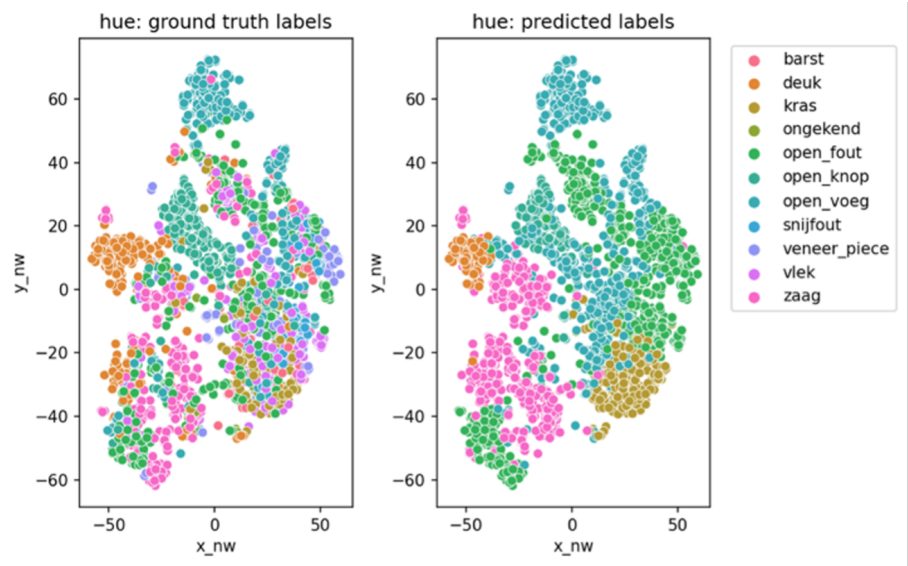
Figuur 4.2: K-means binary clustering

Hoewel mutli-class en binaire K-means clustering niet rechtstreeks vergeleken kunnen worden, blijkt de performantie van binaire clustering veelbelovend, in tegenstelling tot de lage performantie van de mutli-class variant. Dit fenomeen kan worden verklaard door verschillende factoren die voortvloeien uit de inherente verschillen tussen deze twee soorten classificatietaken: bij binaire classificatie wordt de feature space, of de ruimte waarin de kenmerken van de afbeeldingen worden gerepresenteerd, over het algemeen vereenvoudigd omdat het model slechts twee groepen hoeft te onderscheiden. Deze vereenvoudiging kan leiden tot duidelijker gedefinieerde clusters binnen de feature space, waardoor k-means effectiever kan zijn in het toewijzen van afbeeldingen aan de juiste cluster. De resultaten worden samengevat in tabel 4.4.

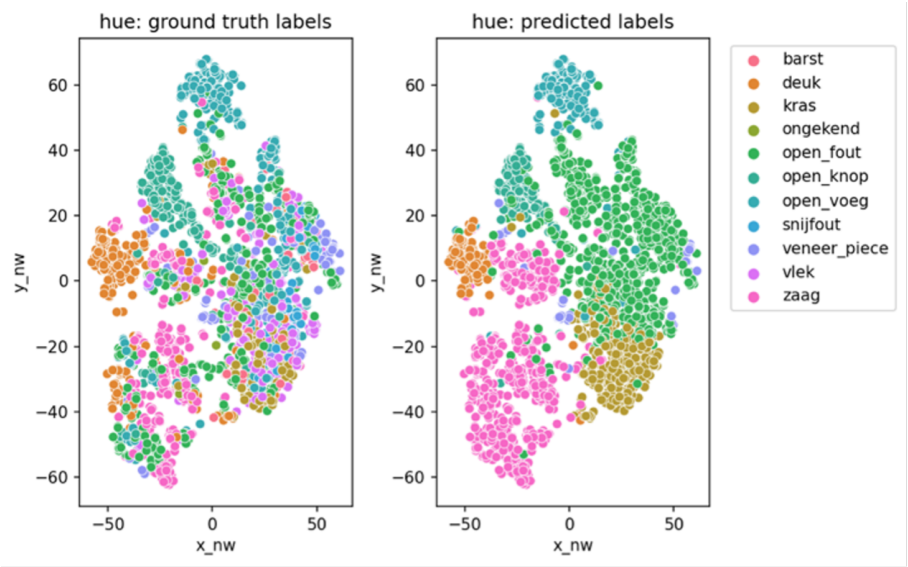
Tabel 4.4 Performantie binaire classificatie.

Classificatie	Backbone	Layer	Accuracy	Precision	Recall
Binair	Vgg 16	Fc 1	0.793	0.869	0.690
Binair	Resnet 50	Fc 1	0.748	0.858	0.594

Effect dimensionaliteitsreductie: Ter illustratie tonen figuur 4.3 en 4.4 het visueel effect van 50 dimensies ten opzichte van 30 dimensies. Tabel 4.12 vat alle testen samen. Deze testen werden uitgevoerd op de eerste fully connected layer van vgg16, zoals voorgesteld door Cohn et al [2]. De dataset blijft dezelfde als in de eerste dataset: alle foutieve windows van de originele dataset.



Figuur 4.3: K-means clustering on 50 dimensions



Figuur 4.4: K-means clustering on 30 dimensions

Tabel 4.5 Accuracy gebaseerd op verschillende dimensies.

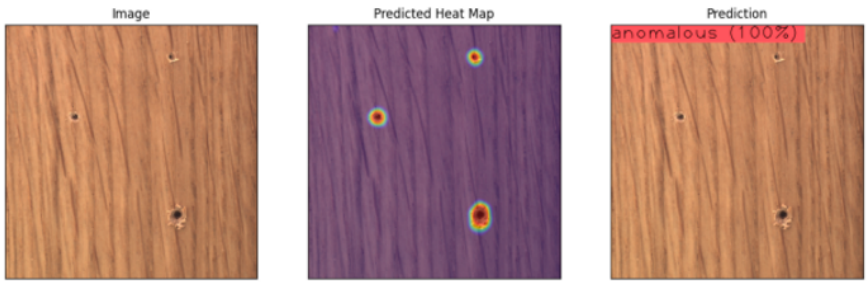
Dimensie	Accuracy
10	0.358
50	0.412
30	0.458
70	0.437
120	0.467

Uit deze resultaten wordt duidelijk dat het reduceren of verhogen van de dimensie geen lineair verband houdt met de accuracy. De trial en error karakteristiek van deze methode wordt duidelijk zoals Cohn et al [2] eerder beschreven. Naarmate de dimensie van de dataset toeneemt, neemt de ruimte binnen elke dimensie exponentieel toe, wat leidt tot het fenomeen bekend als de "curse of dimensionality". Dit kan een negatieve impact hebben op k-means clustering omdat de toename in ruimte ertoe kan leiden dat datapunten gelijkmatig verspreid lijken te zijn, waardoor het moeilijker wordt voor het algoritme om betekenisvolle clusters te vormen. Dimensiereductie kan in sommige gevallen helpen door de relevante informatie in een compactere vorm te concentreren, maar te veel reductie kan ook belangrijke informatie verwijderen die noodzakelijk is voor effectieve clustering.

4.2.2 PaDiM

Het onderzoek naar anomaliedetectie concentreert zich op het PaDiM-model, dat volgens de anomalib GitHub-repository het meest toegepaste model binnen anomalib is. Gezien de aard van anomaliedetectie, beperkt de classificatie zich tot een binaire aanpak.

MVTec AD dataset: Vooraleerst werd als proof of concept het PaDiM algoritme toegepast op de MVTec AD [4] hout defect dataset, waarvan voorbeelden te zien zijn in figuur 4.5. Aangezien PaDiM volgens de literatuur veelbelovende resultaten biedt op deze dataset, bood deze test de kans om dit te verifiëren en te controleren op implementatiefouten. Hier toont tabel 4.6 een hoge performantie. Een opmerking hierbij is dat alle samples zeer gelijkaardig zijn, met een perfecte belichting en weinig tot geen kleurafwijkingen. Ook is er maar één foutklasse, die zeer specifieke, herkenbare features heeft. Hierdoor is het eenvoudiger om een anomalie te detecteren, aangezien de specifieke fout een groot contrast vormt ten opzichte van de background.



Figuur 4.5: inference voorbeeld voor MVTEC AD met PaDiM

Tabel 4.6 Performantie van PaDiM-model op MVTec AD dataset.

Model	Dataset	Accuracy	Precision	Recall
PaDiM	MVTec AD	0.96	1.00	0.90

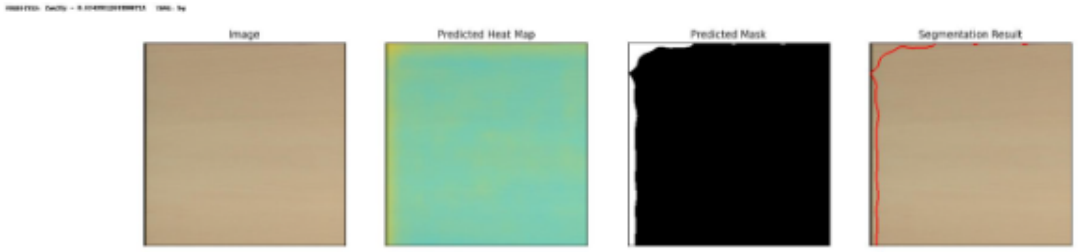
Binaire classificatie op background samples: Vervolgens werd het PaDiM model getraind met background samples uit de train dataset van 4.1. Om de trainingstijd te beperken werd gebruik gemaakt van een random sampling methode, die 3000 background windows kiest. Aansluitend werden testen gedaan met verschillende hyperparameters zoals: treshold, aantal windows, layers, etc. Vervolgens worden binaire performantie metrics gegenereerd, aan de hand van de test dataset, terug te vinden in tabel 4.7.

Tabel 4.7 Performance PaDiM trained on all samples.

Model	# windows	Threshold	Backbone	Layers	Accuracy	Precision	Recall	F2-score
PaDiM	3000	50%	Resnet 18	3	0.33	0.09	0.98	0.33
PaDiM	3000	60%	Resnet 18	3	0.67	0.14	0.83	0.42
PaDiM	500	50%	Resnet 18	3	0.15	0.07	0.99	0.27
PaDiM	3000	50%	Resnet 18	2	0.43	0.10	0.95	0.35
Stfpm	3000	50%	Resnet 18	3	0.08	0.07	1.00	0.27
Fastflow	3000	50%	Resnet 18	3	0.33	0.11	0.96	0.38

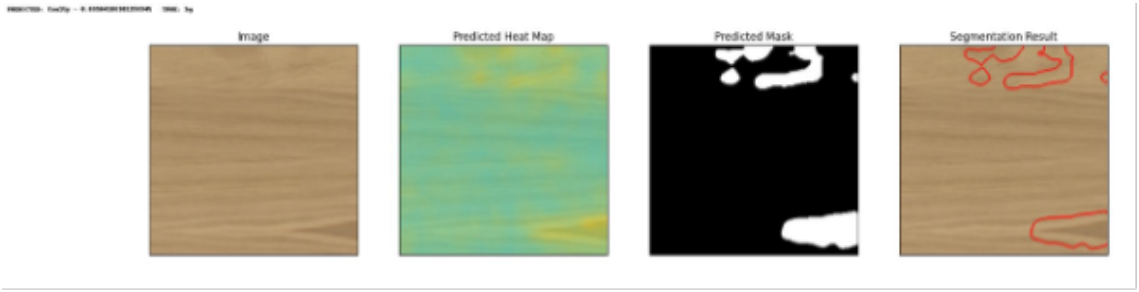
De oorzaak van deze performantie scores wordt onderzocht aan de hand van de visuele weergave van het detectiealgoritme. Bij de eerste test werd de lage accuracy duidelijk. In figuur 4.6 is te observeren dat de rand consistent als incorrect wordt geïdentificeerd, voornamelijk ten gevolge van de abrupte contrastovergang. Deze scherpe overgang past niet

binnen de verwachte normale verdeling, waardoor het systeem deze als afwijkend classificeert.



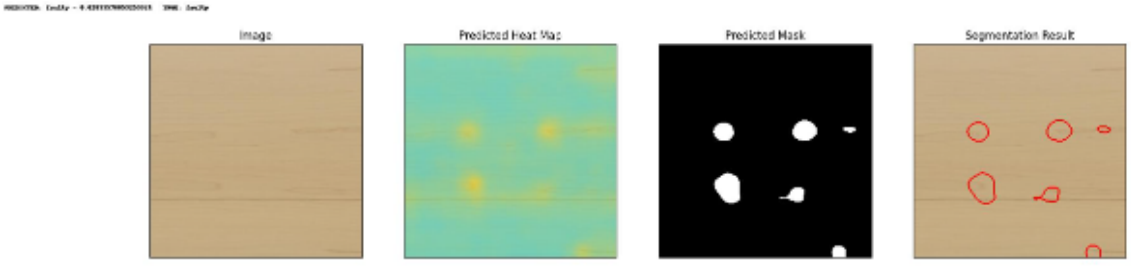
Figuur 4.6: PaDiM inference voorbeeld 1

Figuur 4.7 toont dat patronen eigen aan de houtsoort ook gecategoriseerd kunnen worden als foutief. Dit wil zeggen dat een specifiek patroon ook buiten de gaussiaanse distributie zou vallen.



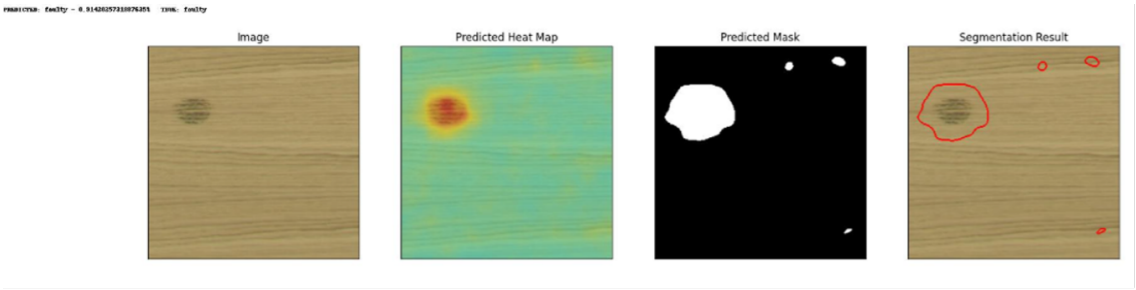
Figuur 4.7: PaDiM inference voorbeeld 2

Figuur 4.8 toont dat het algoritme het ook moeilijk heeft met het detecteren van kleine fouten. De snijlijn die duidelijk zichtbaar is, werd niet gedetecteerd.



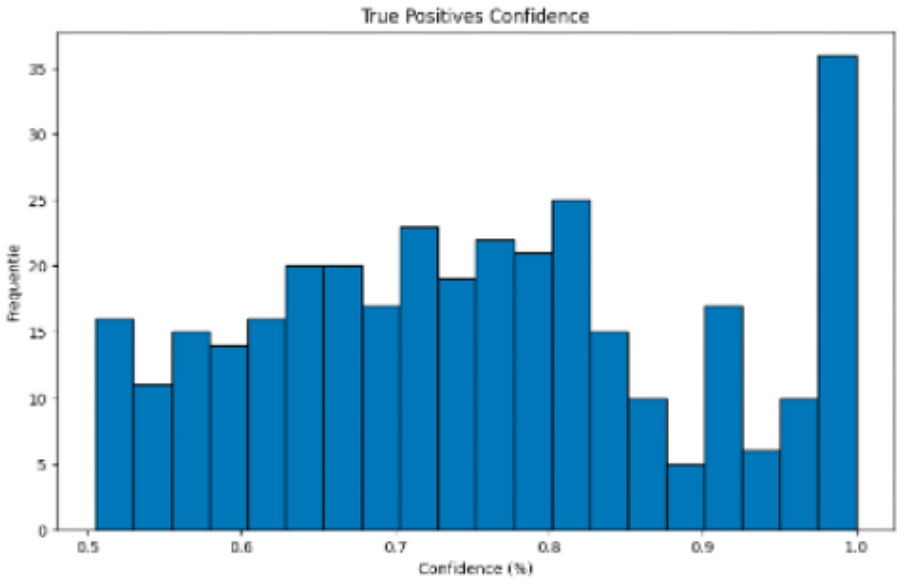
Figuur 4.8: PaDiM inference voorbeeld 3

Explicietere kenmerken, zoals weergegeven in Figuur 4.9, worden echter wel met een hoge mate van nauwkeurigheid gedetecteerd, voornamelijk vanwege de contrastovergang.



Figuur 4.9: PaDiM inference voorbeeld 4

In de literatuur werd duidelijk dat het instellen van een confidence treshold tot betere resultaten kan leiden. Vooraleer dit te doen werden de huidige confidences geanalyseerd. Figuur 4.10 bevat de confidences van de true positive geclassificeerde samples. Terwijl figuur 4.11 de false positive tresholds omvat. Hieruit wordt duidelijk dat de false positives geconcentreerd zijn rond een laag confidence interval [0.5; 0.6]. Dit betekent dat het instellen van een hogere treshold het aantal false positives vermindert. Dit gaat echter ten koste van heel wat true positives, die ook in dit interval gesitueerd zijn. Bij de analyse van de betrouwbaarheidsgrafieken, zoals gepresenteerd in Figuur 4.10 en 4.11, komt naar voren dat een substantieel deel van de 'true positives' zich binnen het interval [0.5; 0.7] bevindt. Dit toont allereerst al aan dat het verhogen van de confidence treshold van 50% naar 60% een negatieve impact zal hebben, omdat het een groot aantal true positives zal veranderen in false negatives. Dit is ook te zien in bovenstaande tabel 4.7, bij een treshold van 60% verlaagt de recall. Dit zorgt echter wel dat de precision verhoogt: er zijn aanzienlijk minder false positives. Toch is er hier geen "noemenswaardige" verbetering waarneembaar.



Figuur 4.10: Confidence interval true positives

Figuur 4.11: Confidence interval false positives

Binaire classificatie op background samples per houtsoort: Zoals eerder vermeld, beschikt Decospan over een uitgebreide collectie houtsoorten, ontwerpen en kleuropties. Om het effect van een model per houtsoort te onderzoeken, werd gebruik gemaakt van de meest vorkomende houtsoort in de Decospan dataset: Europese eik. De training werd uitgevoerd met behulp van het PaDiM-model. Analoog aan voorgaande test werd er enkel met background samples gewerkt, maar nu specifiek gesampled met de geselecteerde houtsoort. Verder werd het effect onderzocht van het wel of niet meenemen van randen in de training. De resultaten van dit onderzoek zijn weergegeven in tabel 4.8. Wanneer er niet getraind werd op randen, werden deze ook uitgesloten in de testdataset.

Tabel 4.8 Performantie model per houtsoort.

Model	Samples	Randen	Thr.	Backbone	Layers	Acc.	Prec.	Rec.	F2
PaDiM 18	250	Ja	50%	Resnet 18	3	0.93	0.31	0.77	0.59
PaDiM 18	250	Nee	50%	Resnet 18	3	0.94	0.42	0.79	0.67

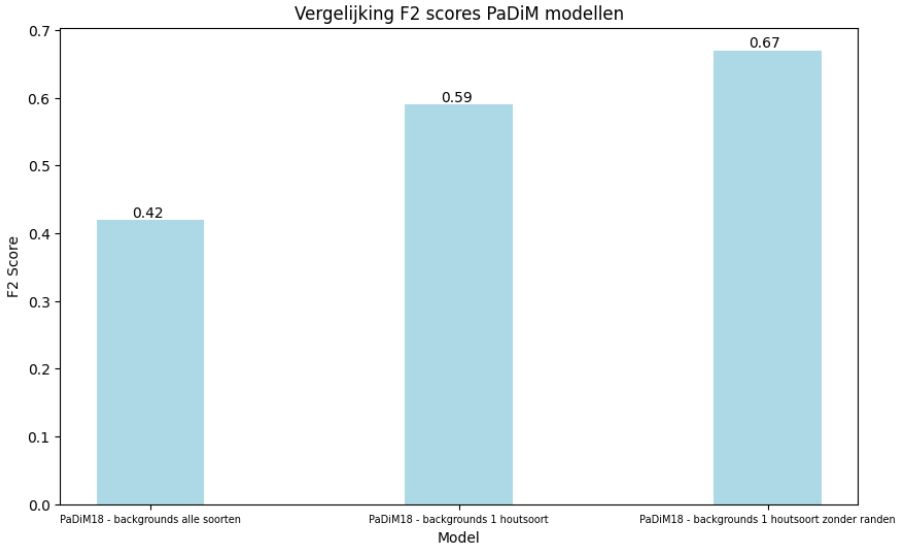
Uit deze resultaten blijkt dat er een verhoogde accuracy, precision en recall waargenomen kan worden bij het trainen op eenzelfde houtsoort. In de test op alle houtsoorten was de hoogste precision 0.14, terwijl deze nu 0.42 is. Nu is er sprake van een hoge accuracy, wat niet het geval was in vorige testen. Dit valt te verklaren omwille van de uniforme houtsoort doorheen de samples: hierdoor is de normale verdeling minder uitgestrekt, wat zorgt dat foutieve samples sneller uit de verdeling vallen. Ook backgrounds worden beter geclassificeerd omdat ze over de samples heen zeer gelijkaardig zijn en passen binnen de normale verdeling. Indien de methode met en zonder randen vergeleken wordt, valt op dat het model zonder randen

betere performantie heeft. Op vlak van precision en recall zijn er noemenswaardige winsten.

4.2.3 Conclusie

Hoewel PaDiM als innovatieve techniek een plaats kan hebben in de industrie, komt het met bijhorende limitaties:
Gevoelig voor "misalignment": Aangezien PaDiM met lokale patches werkt, is het de bedoeling dat het voorwerp steeds exact dezelfde locatie heeft. In datasets zoals MVTec AD performt PaDiM goed, omdat deze dataset met zeer hoge accuratie gecentreerd is. In praktijk is dit minder eenvoudig en vraagt dit meer geavanceerde preprocessing.
Voldoende samples: De PaDiM paper voerde training op 10 000 afbeeldingen uit, zodanig de normale distributie accuraat bepaald kan worden.
Windows moeten groot genoeg zijn: Indien de windows te klein geselecteerd worden bv. 256x256 pixels, generaliseren de patches onvoldoende, waardoor ze nieuwe afbeeldingen die toevallig een karakteristiek van de houtsoort op die patch plaatsen als foutief beschouwen. De normale distributie heeft een grotere regio nodig.
Gevoelig bij randen: Hoewel randen aan de buitenkant zijn, brengen ze de lokale patches in verwarring en raakt de Gaussiaan distributie uit balans, waardoor het ook niet meer in staat is fouten te detecteren. Daarom moeten randen buiten beschouwing gelaten worden.
Gevoelig voor ruis: Jiang et al. [23] merken op dat de effectiviteit van het PaDiM-model vaak te optimistisch wordt voorgesteld in de literatuur, aangezien het gewoonlijk wordt toegepast op data die als 'clean perfect' kan worden beschreven. Deze observatie vindt bevestiging in de context van de MVTec Ad dataset. Huidige modellen voor anomaliedetectie (AD) zijn doorgaans niet ontworpen om om te gaan met 'noisy' data, waardoor hun robuustheid in dergelijke omstandigheden de wensen overlaat. Deze modellen beoordelen samples of patches op basis van hoe sterk ze afwijken van een normale distributie, wat betekent dat de aanwezigheid van ruis de functionaliteit van deze systemen significant kan verstoren. Dit is ook toepasbaar op de Decospan: voornamelijk door de willekeurige patronen in het hout, de randen, de kleurvariaties en de contrasten. Zelfs al wordt er getraind op de houtsoort zelf, blijft de performantie van het model relatief laag, waardoor het niet te gebruiken is in productietoepassingen.

Tenslotte kunnen de beste modellen van de drie testen vergeleken worden op vlak van de F2-score. Grafiek 4.12 toont dat het model getraind op enkel background samples van een specifieke houtsoort, zonder randen, het beste presteert. Hoewel de F2 score van dit model de hoogste is, komt dit met een kost: verhoogde complexiteit. De houtsoortdata dient geïntegreerd te worden in de productielijn, wat een manueel of geautomatiseerd proces kan zijn. Vervolgens, aangezien het model niet getraind is op samples met randen, dient er een detectie-algoritme gemaakt te worden dat samples met randen kan wegfilteren en vervolgens naar een tweede model kan sturen dat fouten op randen kan detecteren.



Figuur 4.12: Vergelijking F2 scores PaDiM modellen

4.3 Binaire CNN

Deze testen proberen een antwoord te formuleren op de onderzoeksvraag "Kan een (supervised) binaire classifier gebruikt worden om de defecten af te zonderen en wat is de performantie tegenover anomaliedetectie?"
De methodologie omvat diverse tests: initieel werd alle beschikbare data direct aan het netwerk geleverd zonder voorafgaande selectie, gevolgd door testen waarbij een sampling methode op de backgrounds werd toegepast om de dataset te balanceren. Vervolgens werd net zoals bij anomaliedetectie onderscheid gemaakt per houtsoort en tenslotte volgt een eigen opgestelde methode, genaamd: de "cut-out" methode, besproken aan de hand van figuur 3.14. Voor deze testen werden volgende hyperparameters toegepast, weergegeven in tabel 4.9.

Tabel 4.9 Hyperparameters binaire CNN

Learning rate	0.001
Batch size	32
Epochs	20
Augmentations	Zie hoofdstuk 3
Criterion	CrossEntropy loss
Optimizer	SGD

Voor elke epoch van de training werd de F2 score berekend en het model met de hoogste

score werd als finaal model voor de test geselecteerd.

4.3.1 Training op alle samples

Bij het trainen van een neuraal netwerk is de meest voordehand liggende en meest gebruikte optie in de industrie, om alle samples te voeden aan een "state-of-the-art" model. Hiervoor werden Yolov8 en Resnet50 gebruikt. Tabel 4.10 geeft het resultaat hiervan weer.

Tabel 4.10 Performantie model getrained op alle samples.

Model	Accuracy	Precision	Recall	F2 score
Resnet50	0.97	0.82	0.14	0.17
Yolov8	0.98	1.00	0.07	0.09

Uit deze resultaten blijkt dat het voeden van alle samples een zeer slechte performantie heeft op vlak van recall voor alle modellen. Dit komt overeen met de conclusie uit hoofdstuk 2.2.4 van de literatuurstudie: indien de dataset niet gebalanceerd is, zal er een inductive bias zijn voor de meerderheidsklasse. Hierdoor wordt er een schijnbaar hoge performantie gehaald op vlak van accuracy, terwijl het model bijna alle samples standaard als background zal karakteriseren.

4.3.2 Training op alle fouten en gesampelde backgrounds

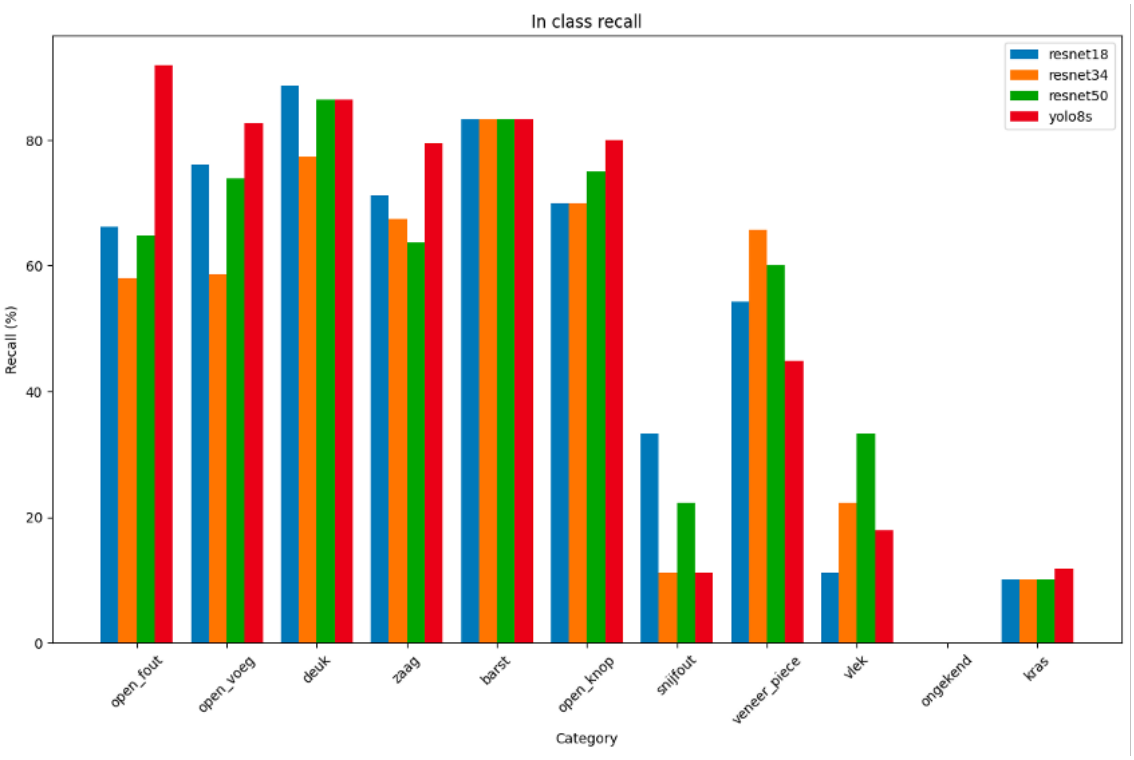
Vervolgens werd een sampling methode toegevoegd, enkel op background windows, met als doel de inductive bias weg te werken. Uit resultaten van tabel 4.11 blijkt hier al een noemenswaardige verbetering plaats te vinden ten opzichte van vorige test. De recall bij Resnet 50 stijgt van 7% naar 62%. Dit bewijst het belang van sampling, zoals aangegeven in de literatuurstudie. De methodologie adresseert het concept van "overfitting". De hypothese dat Resnet50 overfitting zou vertonen en suboptimaal zou presteren, wordt niet bevestigd. Resnet50 vertoont slechts een marginaal verschil van 0,01 vergeleken met het Resnet18-model, terwijl het Resnet34-model inferieur presteert. Dit illustreert dat in machinevisie geen universele oplossing bestaat; elke dataset reageert uniek op verschillende architecturen. De meest plausibele verklaring voor de robuustheid van Resnet18 ligt in zijn generalisatievermogen, wat vooral voordelig is bij complexere datasets zoals die van Decospan, waar generalisatiecapaciteit essentieel kan zijn. Indien Resnet50 en Yolov8s vergeleken worden, is een verlaagde performantie bij Yolo zichtbaar. Hoewel de recall 12% hoger ligt, wordt deze verbetering tenietgedaan met een verlies van 43% in precision. Dit heeft een negatief effect op de f2 score.

Figuur 4.13 is opgesteld aan de hand van de ground-truth annotatie labels. Hoewel het niet verward mag worden met een multi-class classifier, geeft het inzichten over welk type windows goed presteren in de binaire classifier. Twee domeinen spelen een rol: de zichtbaarheid van

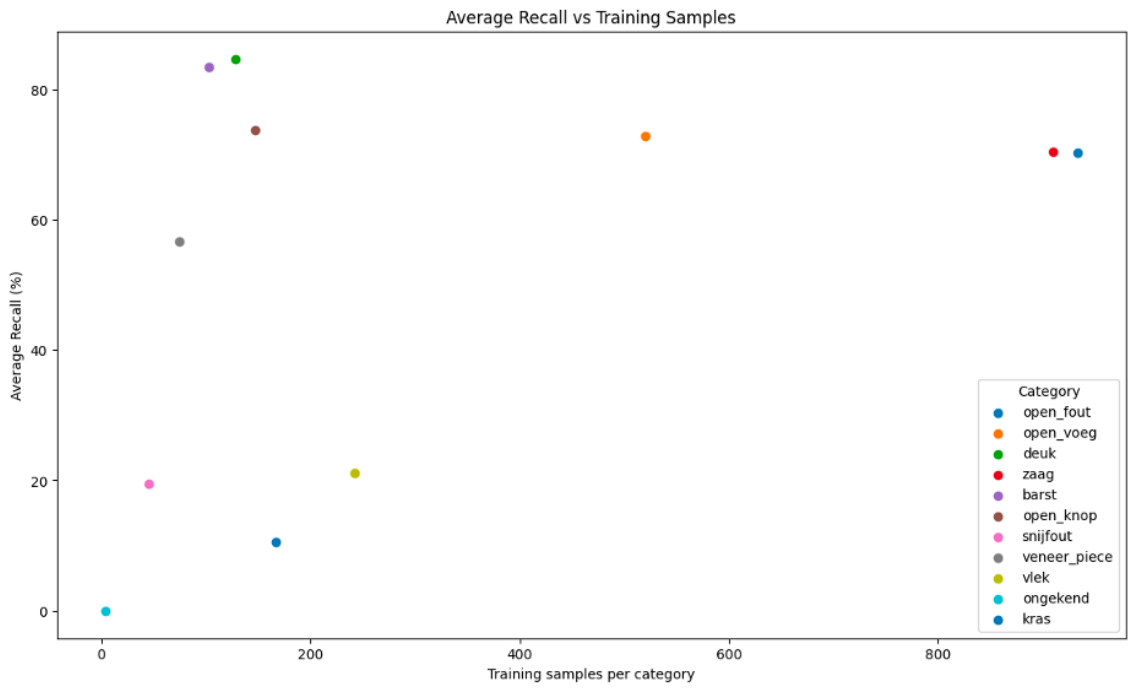
Tabel 4.11 Performantie model getrained op fouten + gesamplede backgrounds.

Model	Accuracy	Precision	Recall	F2 score
Resnet34	0.985	0.686	0.589	0.606
Resnet18	0.984	0.627	0.636	0.634
Resnet50	0.986	0.720	0.628	0.644
Yolov8s	0.950	0.293	0.751	0.572

een fout en het aantal training windows. Zoals verwacht presteren open fout, open voeg en zaag zeer goed, omwille van hun grootschalig aantal samples (500+). Ook open knop, deuk en barst, die slechts rond de 100 samples hebben, halen vergelijkbare prestaties. Om dit te illustreren bevat grafiek 4.14 het aantal samples per categorie op de x-as en de gemiddelde recall over de modellen heen op de y-as.



Figuur 4.13: In class recall voor elk binair model



Figuur 4.14: Average Recall vs Training samples

Uit deze scatter plot blijkt dat het verhogen van het aantal samples niet per se garant staat voor een hogere performantie. Zo heeft de klasse vlek een hoger aantal training samples dan barst, terwijl het een 62% lagere recall heeft. Dit valt te verklaren aan de hand van de visibiliteit van een fout: onderstaande figuren geven een voorbeeld van snijfout, vlek, kras en ongekend. Snijfout, vlek en kras zijn voor het menselijk oog zeer moeilijk detecteerbaar, laat staan voor een neuraal netwerk. Ook is de fout beperkt in grootte en lijkt het zeer goed in te mengen met de patronen van de houtsoort. Hoewel de categorie ongekend wel expliciet van grootte en contrast is, zijn er te weinig samples om het model hierop te laten fitten. Dit wordt visueel weergegeven in figuur 4.15.

Figuur 4.15: Klassen voorbeelden

Het belang van meer samples wordt benadrukt door de open fout categorie, weergegeven in

figuur 4.16. Deze is van nature ook beperkt in grootte en visibiliteit. Door te trainen op voldoende samples slaagt het netwerk er toch in deze categorie met een relatief hoge recall te voorspellen.

Figuur 4.16: Open fault voorbeeld

De 4 vergeleken modellen hebben ieder een andere performantie en reageren verschillend op het aantal getrainde windows. Daarom werd een pearson correlatie coëfficient bepaald van de recall ten opzichte van het aantal samples. Uit tabel 4.12 blijkt dat er een positieve correlatie is voor alle modellen, wat wilt zeggen dat meer samples effectief de recall per klasse verbeteren. Toch blijkt Yolo gevoeliger te zijn voor het aantal samples. De verklaring ligt bij de complexiteit van de taken die het yolo netwerk uitvoert: het doet aan segmentatie, waardoor meer training samples vereist zijn. Dit verklaart ook waarom Yolo de hoogste performantie haalt voor de categorie open fout, deze is de meest vertegenwoordigde klasse met 933 windows.

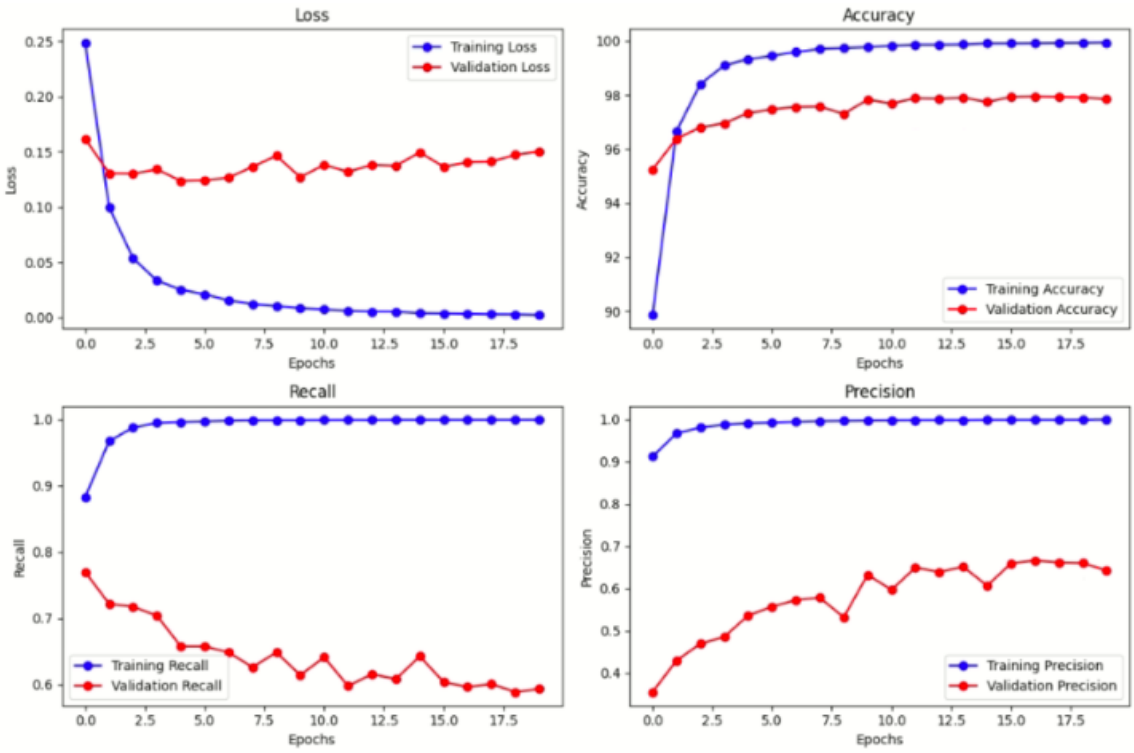
Tabel 4.12 Pearson correlation coefficient.

Model	C-value
Renset18	0.248
Resnet34	0.190
Resnet50	0.190
Yolov8s	0.483

Indien de Resnet backbones onderling vergeleken worden, wordt de generalisatiecapaciteit van Resnet 18 duidelijk. Voor open fout presteert het beter dan Resnet 50, terwijl het voor open knop slechter presteert. Dit omdat open fout geen complexe patronen heeft en hierdoor generaliseerbaar is. Ter illustratie figuur 4.17.

Figuur 4.17: Open class voorbeeld

Tenslotte wordt ook gekeken naar het training proces. Daarin wordt duidelijk dat het uitdagend is voor het model een balans te vinden tussen recall en precision. Grafiek 4.18 bevat het trainingsproces van Resnet18. Hieruit blijkt dat een stijgende precision resulteert in een dalende recall. Hierdoor heeft het trainen van het model voor veel epochs geen nut.



Figuur 4.18: Training metrics

Hoewel de behaalde resultaten veelbelovend zijn, blijft de recall nog steeds beperkt. Deze bevinding onderstreept de noodzaak voor het ontwikkelen van alternatieve, op maat gemaakte oplossingen. Gezien het feit dat Yolov8 als een "out of the box" oplossing beschouwd wordt die minimale customisatie toestaat, voornamelijk vanwege de strikte vereisten van

annotatieformaat, werden de experimenten beperkt tot het Resnet model. Resnet, dat geïmplementeerd is in Pytorch, biedt meer flexibiliteit voor het integreren van aangepaste oplossingen, wat een essentiële factor kan zijn voor het verder verbeteren van de modelprestaties in specifieke toepassingen.

4.3.3 Training per houtsoort

Tabel 4.13 bevat de resultaten van modellen getraind op vlakke houten platen van de houtsoort Europese eik. Dit analoog aan de training in tabel 4.8, maar nu samengevoegd met foutieve windows van de geselecteerde houtsoort.

Tabel 4.13 Performantie model per hout type.

Model	Accuracy	Precision	Recall	F2 score
Resnet34	0.985	0.802	0.603	0.634
Resnet18	0.984	0.705	0.647	0.658
Resnet50	0.986	0.788	0.638	0.663

Ten opzichte van het beste model van vorige testen is er een 3% verbetering waarneembaar in de F2 score ten opzichte van het best presterende model specifiek getraind op een houtsoort. Hierbij dient afgevraagd te worden of deze beperkte verbetering de verhoogde complexiteit van het creëren van meerdere modellen en het "uitzoeken" van de houtsoort tijdens interferentie, waard is.

4.3.4 Training met cut-out methode

Tenslotte werd de cut-out uitgevoerd. Deze testen werden uitgevoerd op dezelfde dataset als hoofdstuk 4.3.2. De resultaten hiervan zijn te zien in tabel 4.14.

Tabel 4.14 Performantie voor cut-out methode.

Model	Accuracy	Precision	Recall	F2 score
Resnet34	0.982	0.781	0.723	0.734
Resnet18	0.989	0.908	0.759	0.785
Resnet50	0.981	0.852	0.594	0.632

Hieruit wordt duidelijk dat de F2 score beduidend hoger ligt dan vorige modellen. Ook is Resnet50, zoals vorige 2 testen, niet meer het beste model. Dit valt te verklaren aan de hand van de cut-out methode. Omdat fouten gecentreerd worden met marge, is het eenduidiger en duidelijker voor het model om te trainen. In vorige testen kon een fout opgesplitst

worden en slechts een beperkt deel van een window innemen aan de randen (bijvoorbeeld de rechterbovenhoek). Hierdoor was een complexer model vereist, zoals Resnet50. Nu het trainingsproces eenvoudiger gemaakt is, volstaat een lichter model, bijvoorbeeld Resnet18 dat beter kan generaliseren. De vraag zou gesteld kunnen worden of deze methode een vertekend beeld van de werkelijkheid geeft. In praktijk is dit echter beperkt omdat er een overlappercentage voorzien is in de SAHI-methode. Zo kan gegarandeerd worden dat er altijd een window zal zijn met de gecentreerde fout. Toch zullen er nog steeds windows aanwezig zijn die een deel van de fout bevatten aan de randen. Stel dat deze dan toch als een background geclassificeerd zou worden, is dit geen probleem, aangezien de window met de gecentreerde versie van dezelfde fout toch als positief geclassificeerd zal worden.

4.3.5 Training op 5 foutcategorieën

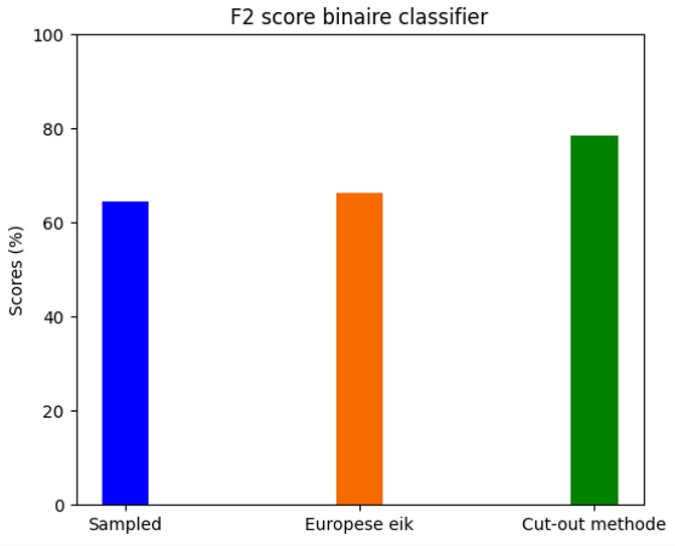
Uit contact met Decospan zelf bleek dat 5 foutcategorieën absolute prioriteit hadden om gedetecteerd te worden: open fout, open voeg, open knop, zaag en vlek. Hierdoor is het interessant om te onderzoeken als een model getraind op minder soorten fouten beter kan presteren en vooral generaliseren. Er wordt een training uitgevoerd met dezelfde dataset als de cut-out methode, omwille van de veelbelovende performantie. Nu zijn de foutieve windows gereduceerd tot de 5 categorieën. De background windows zijn ongewijzigd. Als backbone werd Resnet18 gekozen, omwille van de beste prestatie bij de cut-out methode uit vorig hoofdstuk. Uit tabel 4.15 blijkt er geen betere performantie uit deze methode voort te vloeien. De accuracy blijft gelijk en de F2 score daalt een 2%. Hierdoor kan besloten worden dat voor binaire classificatie het aantal samples van eender welke fout belangrijker zijn voor de performantie van het model.

Tabel 4.15 Performantie voor cut-out method getrained op 5 categorieën.

Model	Accuracy	Precision	Recall	F2 score
Resnet18	0.989	0.875	0.741	0.763

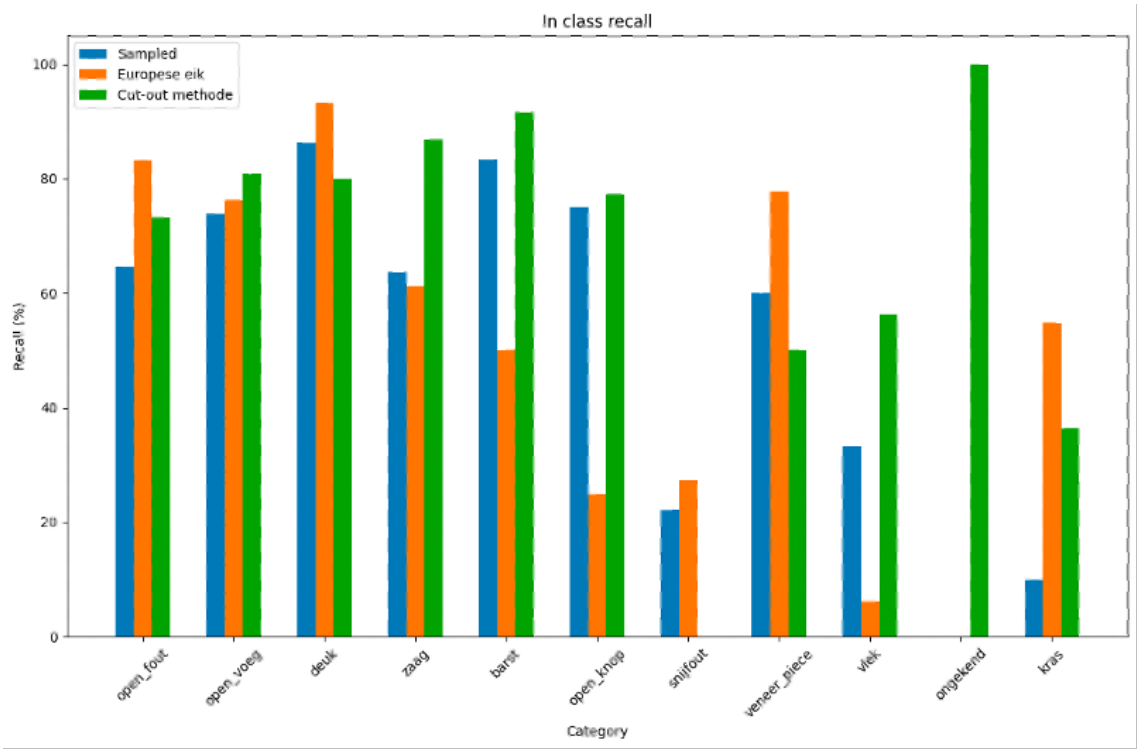
4.3.6 Conclusie

Figuur 4.19 bevat een vergelijking van de modellen met de hoogste F2 score van de 3 methoden. Hieruit blijkt dat de cut-out methode het best presteert met een F2 score tot 12% hoger.



Figuur 4.19: F2-score vergelijking

Op grafiek 4.20 worden de drie methoden vergeleken per foutcategorie.



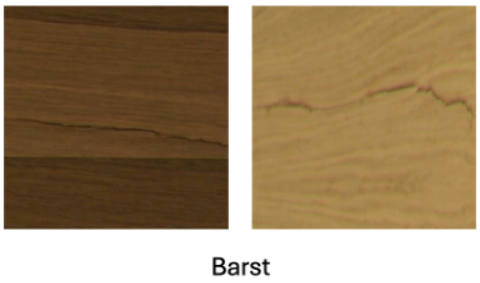
Figuur 4.20: Recall vergelijkig per klasse

Hieruit blijkt dat de methode van een model per houtsoort superieur presteert op fouten met een grote variatie per houtsoort. Ter illustratie figuur 4.21, een voorbeeld voor de open fout categorie, omdat deze fout een drager, die niet volledig bedekt is, categoriseert, verschillen

de features van deze fout per houtsoort. Hierdoor presteert een neuraal netwerk specifiek getraind op de houtsoort beter.

Figuur 4.21: Class voorbeeld

Echter bij categorieën waar de samples gelijkaardig zijn qua features door de houtsoorten heen, presteert de methode van een model per houtsoort minder goed, omwille van het verminderd aantal samples. Ter illustratie figuur 4.22, de categorie barst behoudt zijn structuur en vorm doorheen de houtsoorten. Hierdoor is het verhogen van het aantal samples efficiënter dan een apart model voorzien.



Figuur 4.22: Class voorbeeld

Algemeen kan besloten worden dat het aanmaken van een model per houtsoort kan zorgen voor een superieure performantie in bepaalde klassen. Echter is dit in het geval van Decospan geen realistische keuze, omdat er 150 houtsoorten zouden zijn. Dit zou de ontwerptijd van het algoritme enorm verhogen en zou onpraktisch zijn, omdat er een methode moet gemaakt worden om de houtsoort op de productielijn te communiceren. Indien de "sample" en "cut-out" methode vergeleken worden, kan besloten dat de "cut-out" methode de beste performantie heeft in het merendeel van de klassen. Hierdoor is dit de geprefereerde methode om op verder te werken.

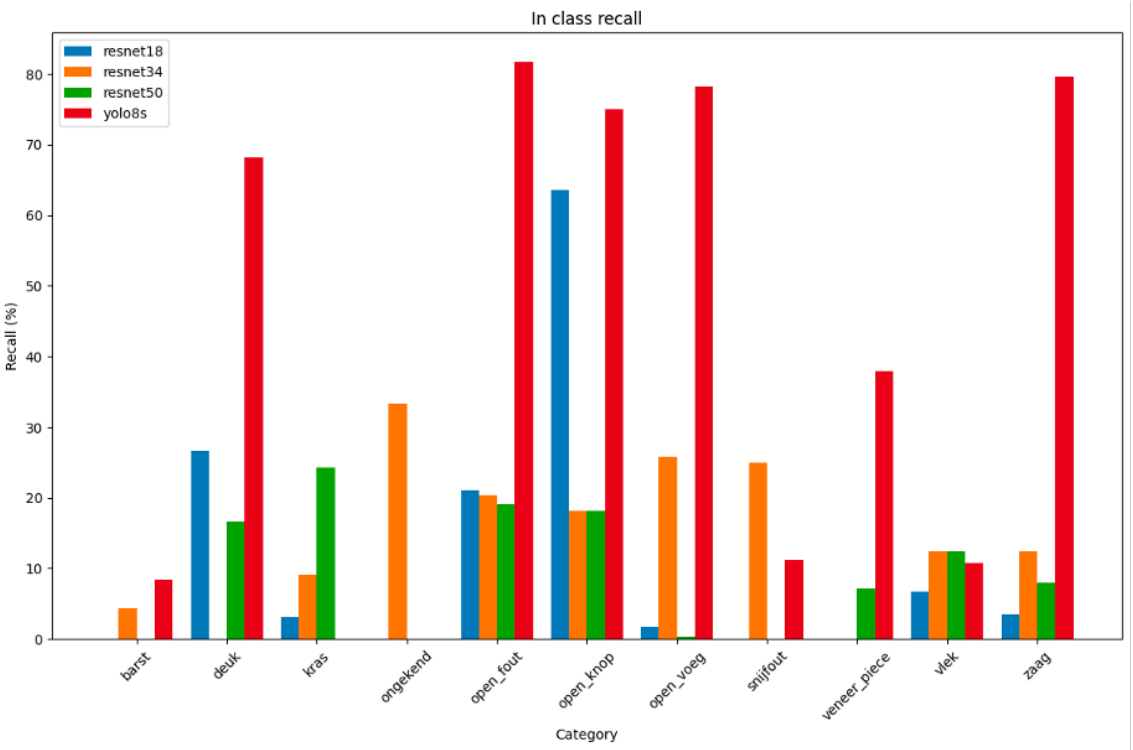
4.4 Multi-class CNN

In de ontwikkeling van de multi-class CNN-aanpak is gekozen voor een voortzetting van de "cut-out methode" voor het gebruik van Resnet, vanwege de superieure prestaties die deze methode biedt. Voor Yolo is verdergegaan met de klassieke SAHI-methode. De performantie van twee approaches wordt vergeleken. Enerzijds de klassieke approach: het trainen van een model op alle foutieve windows en een evenredig aantal gesampelde backgrounds, afkomstig uit tabel 4.1. Anderzijds het trainen met enkel de foutieve windows. De bijgevoegde tabellen presenteren de interferentieresultaten van de testdataset.

4.4.1 Training op fouten en backgrounds

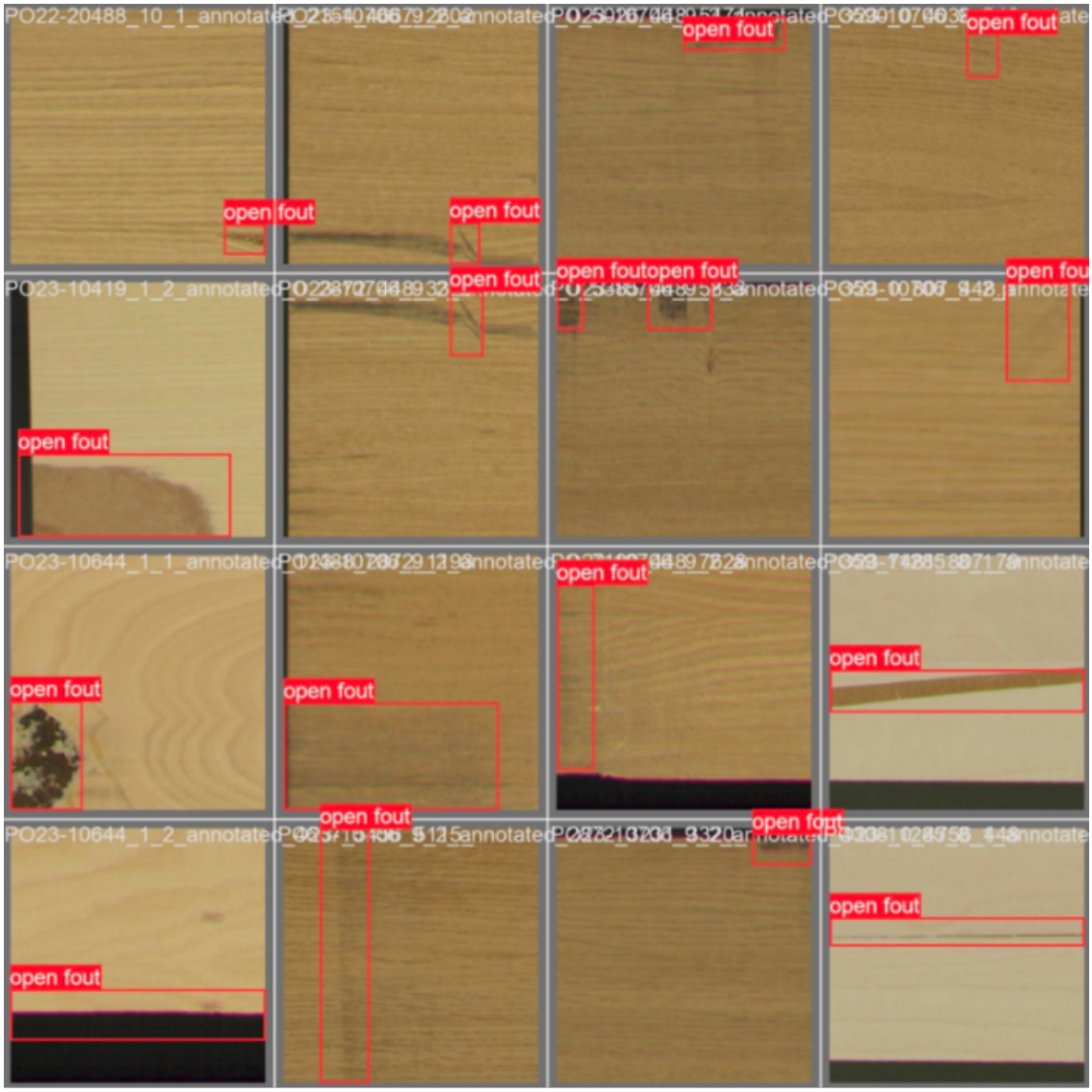
Tabel 4.16 Performantie per model, getraind op fouten en gesampelde backgrounds

Model	Accuracy
Resnet34	0.039
Resnet18	0.038
Resnet50	0.064
Yolov8s	0.953

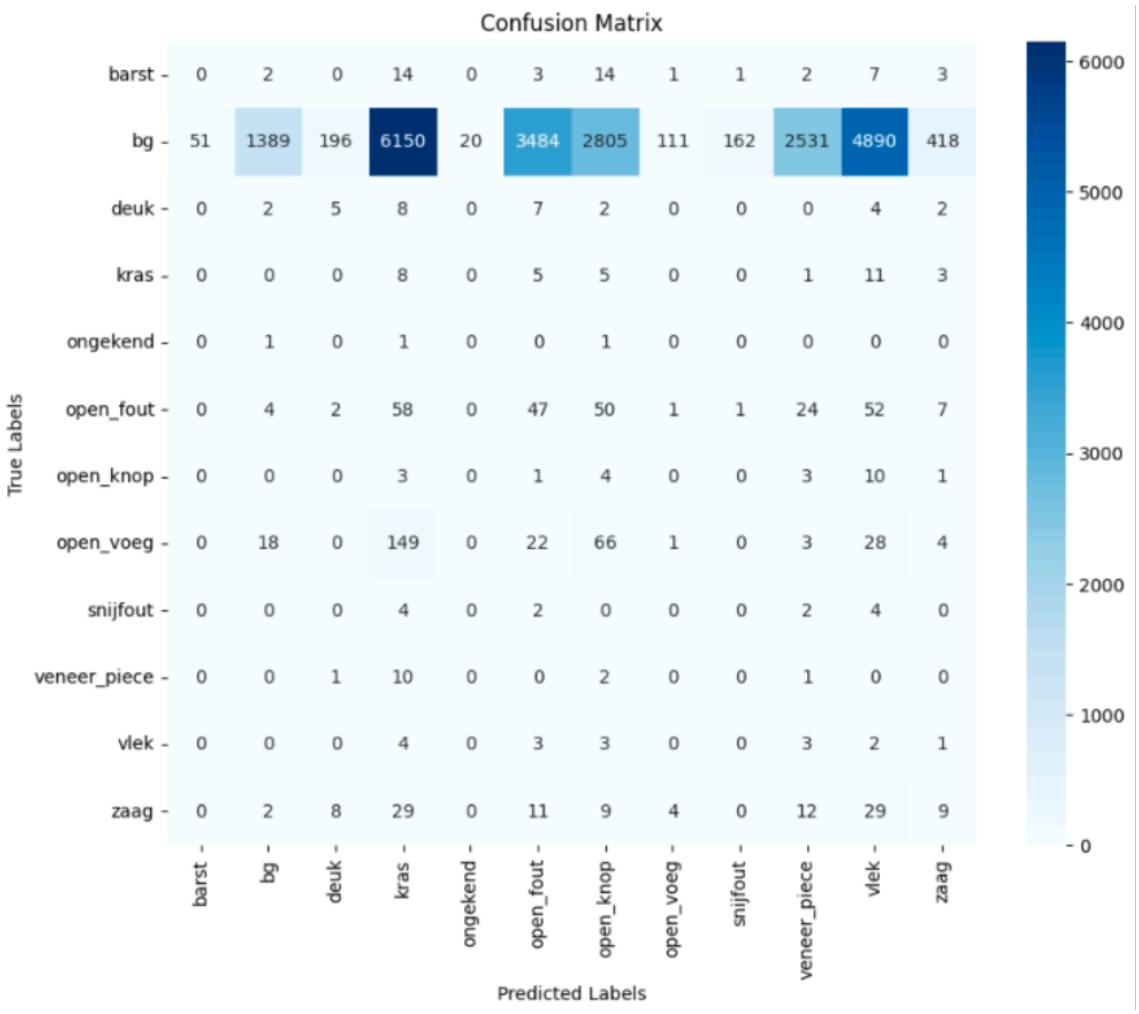


Figuur 4.23: In class recall voor elk model

Uit tabel 4.16 en grafiek 4.23 wordt duidelijk dat Yolov8s superieure performantie heeft. Dit valt te verklaren door het aanwezig zijn van meerdere klassen. Voorheen werd er aan binaire classificatie gedaan. Nu zijn er 11 klassen met een bijkomende background klasse. Hierdoor is het voor ieder model complexer om fouten juist te classificeren. Hierdoor komt de sterkte van Yolo naar voren. Omdat het niet enkel classificeert maar ook segmenteert, localiseert het eerst de fout en probeert dan pas de klasse in de schatten op dat specifieke deel van de windows. De Resnet modellen moeten een klasse inschatten op basis van een volledig window, wat complexer is, omdat er ook een groot background gedeelte aanwezig is. Figuur 4.24 toont een voorbeeld van de segmentatie, hierdoor kan Yolo veel gerichter classificeren. De lage accuracy van de Resnet modellen kan verklaard worden aan de hand van de confusion matrix: omdat de testdataset een grootschalig aantal backgrounds heeft, wordt de accuracy metric grotendeels bepaald door hoe goed het model in staat is backgrounds juist te classificeren. Uit figuur 4.25, een voorbeeld van Resnet18, blijkt dat Resnet modellen er slechts beperkt in slagen de background windows juist te classificeren.

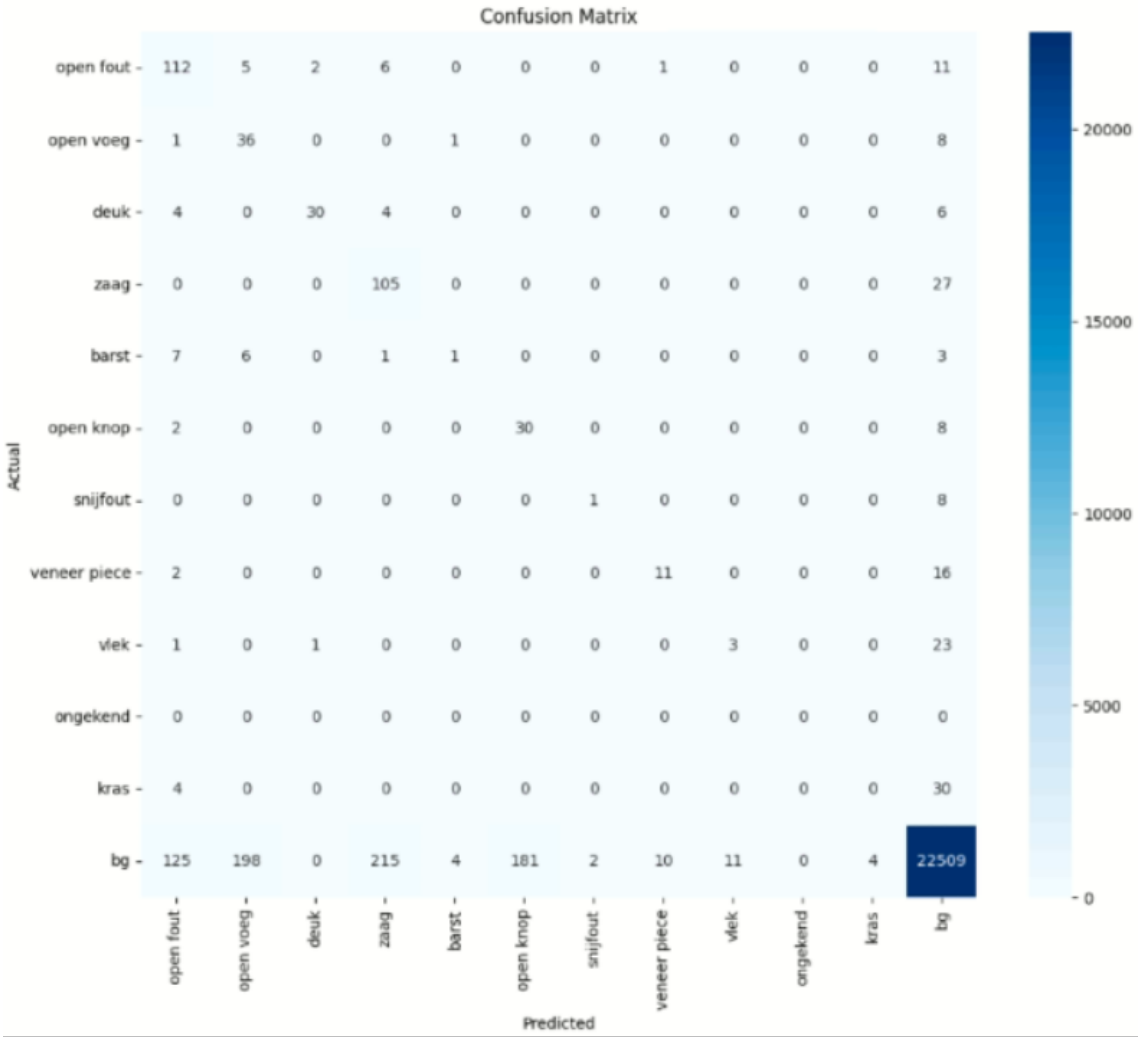


Figuur 4.24: Voorbeeld van segmentatie



Figuur 4.25: Confusion matrix Resnet18

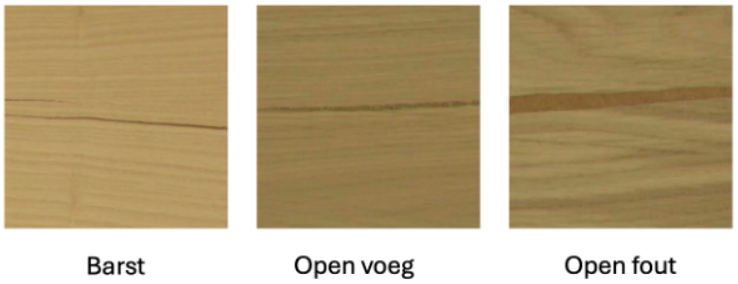
Indien verder ingegaan wordt op Yolov8s, kan er analoog aan de binaire classificatie, een sterke performantie voor de open fout, open voeg, deuk, zaag en open knop categorie waargenomen worden. Als er naar de confusion matrix, figuur 4.26, van yolov8s gekeken wordt, is er algemeen een trend zichtbaar: voorspelde klassen worden consistent verward als open fout. Dit allereerst omwille van de bias naar de open fout toe: het is de meest voorkomende klasse met 933 samples, maar ook omwille van gedeelde kenmerken. Een open fout heeft als kenmerk dat de bovenste laag van het hout voor een deel verdwenen is, waardoor de drager zichtbaar is. Dit kenmerk wordt gedeeld door andere foutklassen, waardoor er verwarring is. Figuur 4.27 is in realiteit een deuk, terwijl deze door het zien van het onderliggend houtpatroon geclassificeerd wordt als een open fout.



Figuur 4.26: Confusion matrix Yolov8s

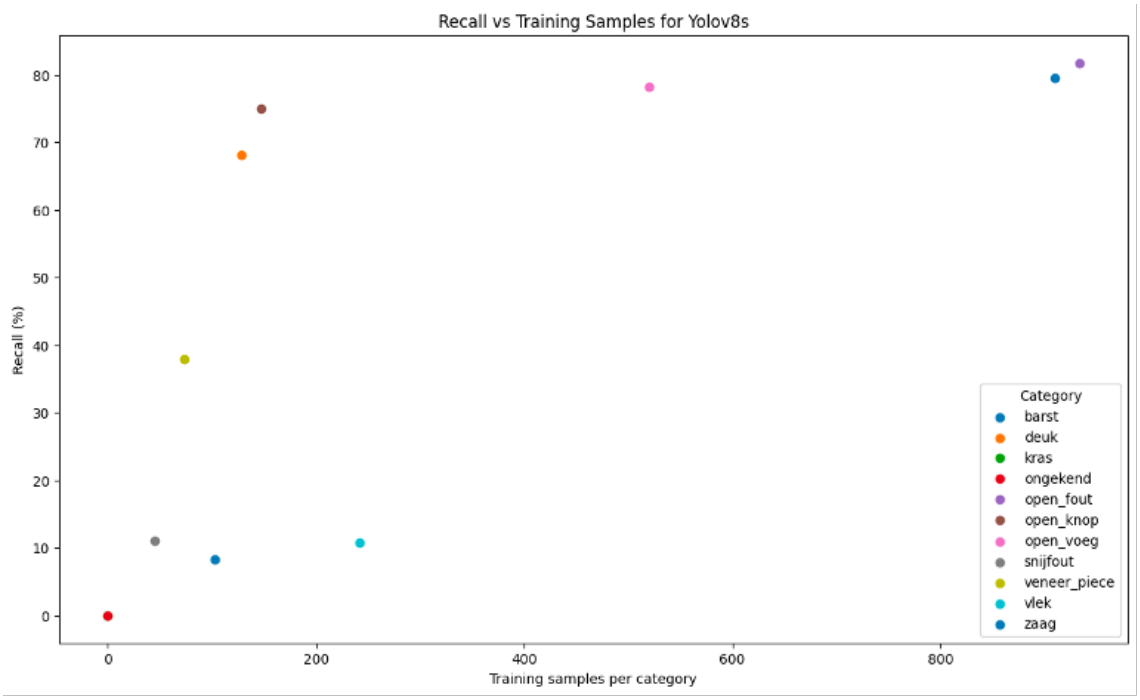
Figuur 4.27: Segmentatie voorbeeld

Wat opvalt, is dat de performantie voor barst minder dan 10% is. Dit in tegenstelling tot de binaire classificatie waar windows met ground truth "barst" goed presteerden met een recall van meer dan 80% voor het opdelen in een van de twee categorieën. Dit omwille van verwarring met de open fout en open voeg categorieën. 38% van de tijd wordt het geclassificeerd als open fout, 33% van de instanties als open voeg. Dit omwille van de karakteristiek, hoewel er bij barst effectief een barst is en niet enkel sprake is van het verwijderen van de bovenste laag, wekt het wel de andere indruk. Er is sprake van een visuele overeenkomst tussen de 3 categorieën, wat het voor Yolo moeilijker maakt onderscheid te maken. Dit verklaart waarom "barst" wel goed presteerde in binaire classificatie, omdat het het patroon als foutief analyseerde, maar niet in multi-class, omdat er effectief een categorie moet toegewezen worden. Als voorbeeld figuur 4.28.



Figuur 4.28: Categorie voorbeeld

Indien een scatterplot gemaakt wordt, zie figuur 4.29, voor het Yolo model, kunnen dezelfde conclusies als bij de binaire classificatie genomen worden. Het voorzien van meer samples verbetert de performantie, maar ook de visibliteit van de fout is van belang. Ook hier blijkt dat zaag en vlek geen hoge performantie hebben, terwijl ze wel meer dan 100 samples ter beschikking stellen voor het trainproces.



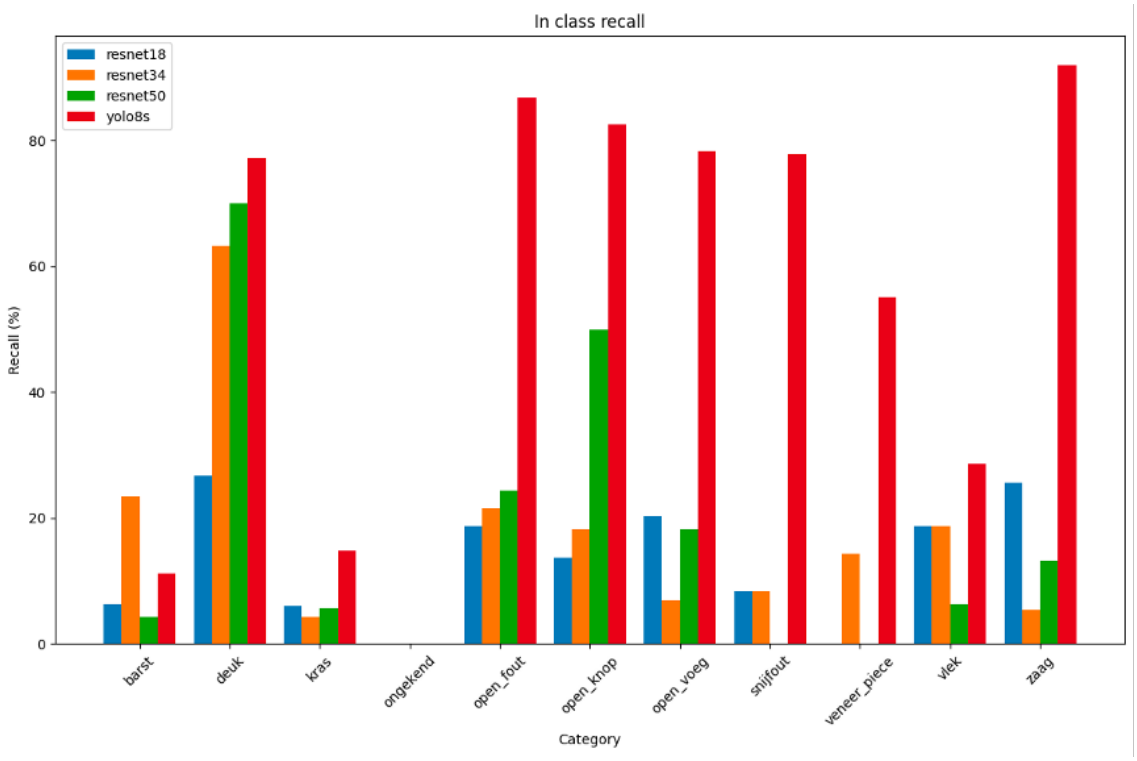
Figuur 4.29: Recall vs training samples voor Yolov8

4.4.2 Training op enkel fouten

Indien enkel getraind wordt op fouten, blijft de filosofie hetzelfde: het Yolo model presteert beter omwille van het segmentatie deel. Uit tabel 4.17 en grafiek 4.30 wordt duidelijk dat het Yolo model nog beter presteert indien er enkel op fouten getraind wordt. De performantie van Resnet is gestegen met meer dan 10%, wat ook aantoont dat het verwijderen van de background klasse positieve gevolgen heeft voor de performantie.

Tabel 4.17 Performantie per model, getraind op enkel fouten

Model	Accuracy
Resnet34	0.146
Resnet18	0.186
Resnet50	0.199
Yolov8s	0.727



Figuur 4.30: Recall vergelijking per klasse

4.4.3 Training op 5 foutcategorieën

Zoals besproken in hoofdstuk 4.3.5, wordt ook voor het multi-class model onderzocht of er een performantieverbetering plaatsvindt als er op minder categorieën getraind wordt. Ook hier werd dezelfde dataset gebruikt, minus de fouten die niet tot de 5 categorieën behoren. Er werd enkel op fouten getraind, omdat dit de beste performantie biedt volgens voorgaande testen. Tabel 4.18 toont aan dat er slechts een performantieverbetering van rond de 2% plaatsvindt, wat zorgt dat deze methode niet als efficiënt beschouwd kan worden.

Tabel 4.18 Performantie per model, getraind op enkel fouten uit 5 categorieën

Model	Accuracy
Yolov8s	0.751

4.5 Praktische implementatie

Uit de vereisten van Decospan kwam voort dat de maximum inferentietijd per plaat 3 seconden is (na inlezen linescan camera). Om deze vereiste te halen, dient elke stap van de pipeline in kaart gebracht te worden, weergegeven in figuur 4.31.

Figuur 4.31: Praktische pipeline

Hierbij wordt er gestart met het SAHI-algoritme. Deze splitst de bronafbeelding op in meerdere windows. De gemiddelde width van de afbeeldingen in de dataset is 10126 pixels en de gemiddelde height is 5592 pixels. Dit wil zeggen dat wanneer er windows gemaakt worden van 448x448 pixels met een overlap van 20%, er gemiddeld 464 windows per afbeelding zijn. Indien er naar de bovengrens gekeken wordt, is een afbeelding maximaal 15879 pixels in breedte en 7949 in hoogte. Dit levert maximaal 924 windows op. Hierdoor stijgt de complexiteit van de probleemstelling, aangezien er in het slechtste geval inferentie op 924 windows binnen de 3 seconden dient te gebeuren. Voor de inferentie van windows zijn er drie belangrijke stappen: allereerst dient preprocessing uitgevoerd te worden op de afbeeldingen (normalisatie, resizing en omzetting naar tensor formaat). Vervolgens moet de afbeelding verstuurd worden naar het video memory van de GPU. Tenslotte volgt de effectieve inferentie op de GPU. Testen werden uitgevoerd op twee verschillende systemen: een Alienware desktop pc en een Jetson Nano Orin. Tabel 4.19 vergelijkt de specificaties van beide systemen, teruggevonden via de systeeminformatie geleverd door de besturingsystemen. Enerzijds wordt theoretisch, ruw geschat, verwacht dat de alienware 5,9x sneller is voor GPU prestaties, omwille van het hoger aantal GPU cores en kloksnelheid. Anderzijds is de Jetson een aantrekkelijkere optie voor de industrie, omwille van de compacte vorm en de relatief lage kostprijs.

Tabel 4.19 Vergelijking van Alienware and Jetson specificaties.

	Alienware	Jetson
Processor	AMD 1950X	Arm Cortex-A78AE
CPU cores	16 core	6 core
CPU clockspeed	3400 MHz	1500 MHz
RAM	32 GB	8 GB (shared)
GPU	NVIDIA GTX 1070 TI	NVIDIA Ampere architecture GPU
GPU cores	2432	1024
GPU clockspeed	1607 MHz	625 MHz
GPU Bandwidth	256 GB/s	68 GB/s
GPU memory	8 GB	8 GB (shared)

Uit deze tabel wordt duidelijk dat de Jetson een beperkt aantal GPU en CPU cores heeft, met een verminderde kloksnelheid. In combinatie met het shared memory tussen de GPU en

de CPU lijkt deze optie minder aantrekkelijk dan de Alienware. Hierdoor is de batch size in testen ingesteld op 32 windows. Een hoger aantal liet de kernel van de Jetson crashen.

Tabel 4.20 bevat resultaten van inferentie op beide platformen. Hiervoor zijn optimalisaties uitgevoerd om zoveel als mogelijk te parallelliseren en gebruik te maken van de CUDA architectuur. Als extra optimalisatie werden afbeeldingen en bewerkingen op afbeeldingen nooit naar de disk geschreven, maar enkel in het ram opgeslaan. Om realistische resultaten te bekomen, werd elke test 10x uitgevoerd en werden de gemiddelde tijdsduren genomen.

Tabel 4.20 Performantie vergelijking tussen Alienware en Jetson.

	Alienware [ms per window]	Jetson [ms per window]
SAHI-algorithm	0.275	0.344
Preprocessing	0.826	0.551
Download to GPU	0.129	0.131
Inference Resnet18	0.118	0.278
Inference Resnet34	0.195	0.472
Inference Resnet50	0.271	0.634
Inference Yolov8s	8.54	33.61

De puur CPU gerelateerde taak uit deze tabel, is het SAHI algoritme. Hier is voor de Alienware een performantiewinst te zien van 20% ten opzichte van de Jetson. Een verklaring hiervoor is het verhoogde aantal cores en de kloksnelheid. Echter zou de theoretische verwachting een stuk hoger liggen. Hiervoor moet gekeken worden naar de implementatie van de SAHI library. Deze is niet geoptimaliseerd voor de specifieke architectuur van de CPU en kan hierdoor niet ten volle gebruik maken van de architectuur. Dit verklaart het beperkte verschil tussen de Alienware en de Jetson. Deze analogie kan ook verder gezet worden naar de GPU taken. Een opvallende parameter is de preprocessing tijd. Deze blijkt voor de alienware hoger te liggen. Ook hier komt dit omwille van de implementatie van de library. Voor de Jetson is een speciale Pytorch versie gecompileerd die door Nvidia en de makers van PyTorch geoptimaliseerd is voor de AI chip van de Jetson en het bijhorende platform. Voor de Alienware is dit niet beschikbaar. Dit trekt zich ook door in de inferentietijden van de modellen. Hoewel de Alienware beter presteert, is de inferentietijd voor bijvoorbeeld Resnet 18 slechts 57% lager dan de Jetson. Terwijl dit niet overeenkomt met de theoretische schatting van 5.7 keer lager. Dit toont aan dat optimalisatie voor specifieke AI chips een competitieve performantie kan geven ten opzichte van algemen GPU's. Anderzijds wordt de omgekeerde situatie bij Yolo aangetoond: deze is niet geoptimaliseerd voor het Jetson platform, waardoor de betere hardware van de Alienware zorgt voor een veel hogere performantie. De inferentietijd ligt 3.94 keer lager.

In voorgaande hoofdstukken bleken volgende modellen de beste performantie te hebben en zullen gebruikt worden als maatstaf voor berekeningen in dit hoofdstuk:

- Binaire classificatie: Resnet18
- Multi-class classificatie: Yolov8s

Tenslotte is ook een piste onderzocht, gebaseerd op het onderzoek van Shi et al. [1], waar eerst een binaire classifier geplaatst wordt, gevolgd door een multi-class classifier. De implementatie van de tweede stage heeft 2 mogelijkheden:

1. Louter werken als analysetool. In dit geval is de tweede stage geen onderdeel van de realtime pipeline. De resultaten komen nadien binnen, wat mogelijk geen probleem is voor Decospan die enkel realtime binaire classificatie wilt uitvoeren.
2. Enkel het verwerken van foutieve windows, geleverd door het binaire classificatie model. Hierdoor moeten er maar een beperkt aantal windows verwerkt worden, wat zorgt dat het model mogelijks nog binnen de realtime constraint valt.

Als referentie wordt er gekeken naar het worst case scenario waar er inferentie dient te gebeuren op 924 windows. Voor de laatste rij is een praktisch scenario voorzien voor de 2 stage approach, waar 5% van de windows als foutief beschouwd worden. Hierdoor kan bekeken worden of mogelijkheid 2 van vorig punt effectief haalbaar is. De resultaten hiervan zijn terug te vinden in tabel 4.21.

Tabel 4.21 Inference tijd vergelijking tussen Alienware and Jetson.

Type	Gemiddelde inferentietijd per window [ms]		Totale inferentietijd voor 924 windows [s]	
	Alienware	Jetson	Alienware	Jetson
Binaire classificatie	1.348	1.304	1.245	1.204
Multi-class classificatie	9.770	34.636	9.027	32.003
2 model approach stage 1	1.348	1.304	1.245	1.204
2 model approach stage 2	8.540	33.610	7.890	31.055
2 model approach in practice	1.775	2.985	1.640	2.757

Uit deze tabel wordt allereerst duidelijk dat de multi-class classificatie met Yolo niet voldoet aan de realtime constraint op beide platformen, terwijl de binaire classificatie dit wel doet. De binaire classificatie met Resnet18 valt ruim binnen de tijdconstraint op beide platformen. Indien naar de 2 model approach gekeken wordt, is de mogelijkheid voor piste 1 realistisch op beide platformen. Enkel stage 1 dient realtime te zijn en valt ruim binnen de grenzen. Indien piste 2 bekeken wordt, wordt duidelijk dat het Yolo algoritme de bottleneck voor de Jetson is. Met 2.75 seconden bereikt dit vrij dicht de grens van 3 seconden. De Alienware heeft dit probleem niet, omdat deze superieure performantie heeft voor Yolo inference.

4.6 Financiële analyse

Een financiële analyse oogt een inschatting te maken van de kost van een investering en het bijhorende rendement. Toegepast op Decospan omvat dit samengevat het berekenen of de kost van het implementeren van dergerlijk AI-systeem het waard is ten opzichte van de uitval van platen.

Hiervoor wordt in tabel 4.22, gestart met basis informatie over het financiële aspect van het productieproces, op basis van data geleverd door Desmet et al. [24]. Deze werden bepaald voor een gemiddeld paneel van 3,5 kubieke meter met gemiddelde data uit het boekjaar 2022 en 2023.

Tabel 4.22 Kosten en productiegegevens van platen. [24]

Beschrijving	**Waarde**
Productieprijs plaat	€ 105
Verkoopprijs plaat	€ 175
Totale productie	841 376
Uitval	4 627
Percentage uitval	0.5%

Uitval kan opgesplitst worden in 2 categorieeën: gedetecteerd door de klant en gedetecteerd door de operator aan de laatste stage van het productieproces. Hierbij worden enkel technische fouten beschouwd. Een klant kan ook retourneren omwille van esthetische redenen. Uit tabel 4.23 blijkt dat de operator in staat is 71% van de technische fouten te detecteren.

Tabel 4.23 Detectieratio's door operators en klanten. [24]

Detectie	**Aantal**	**Percentage**
Operator	3275	70,78%
Klant	1352	29,22%

Vervolgens is het belangrijk om in te gaan op de gevolgen wanneer de klant een fout in een plaat detecteert.

Allereerst dient er een keuze gemaakt te worden of terugsturen de moeite is qua kosten/baten.

- De klant retourneert het paneel
- De klant mag het paneel houden, omdat retourneren economisch niet interessant is

Dan is er nog het financiële gevolg:

- De klant ontvangt een nieuw of hersteld product, zonder meerkost
- De klant krijgt het geld terug en bestelt niet opnieuw bij Decospan

Indien deze 2 opsommingen gecombineerd worden, zijn er 4 mogelijke scenario's, weergegeven in tabel 4.24.

Tabel 4.24 Scenario's van retourzendingen en terugbetalingen. [24]

Scenario	Retour	Terugbetaling
1	Ja	Nee
2	Ja	Ja
3	Nee	Nee
4	Nee	Ja

Echter worden enkel scenario 1 en 2 beschouwd, omdat Decospan altijd opteert voor een retour. Enerzijds om misbruik te vermijden, anderzijds omdat de plaat mogelijks hersteld kan worden. Voor het berekenen van de bijkomende kost per scenario is er een rechtstreekse vergelijking met het scenario waar een fout ter plaatste gedetecteerd wordt. In dat geval zijn de grondstofkosten en productiekosten ook verloren (of de reparatiekosten) en dient er een nieuwe plaat geproduceerd te worden (tenzij de klant geen nieuwe plaat wenst). Hierdoor zijn deze kosten steeds aanwezig in het geval van een fout en dus niet uniek aan een scenario waar de klant een fout ontdekt. Een belangrijke opmerking hierbij is dat de platen gemiddeld in bulk per 10 verstuurd worden. Daarom bevat tabel 4.25 de kosten voor 10 platen, omdat deze volgens Decospan typisch alle 10 defect zullen zijn. Vervolgens worden volgende assumpties gemaakt:

- Verzendkosten: €110
- Retourkosten: €165
- Administratiekosten: €115

Om volledig te zijn, zou ook een inkomst voor het recupereren van het materiaal in rekening gebracht kunnen worden. Decospan geeft aan dat dit buiten beschouwing gelaten kan worden, omdat bestellingen steeds op maat zijn en het niet mogelijk is te recupereren voor andere klanten. Daarom wordt steeds de mogelijkheid voor herstelling bekeken. Tenslotte wordt voor scenario 2 ook nog de gemiste inkomsten in rekening gebracht, omdat de klant niet voor inkomsten gezorgd heeft.

Tabel 4.25 Overzicht van scenario's en bijbehorende extra kosten door klant. [24]

Scenario	Oorzaak bijkomende kost	Kost
1	Retourzend- en verzendkosten + administratie	€ 390
2	Retourzendkosten en 10 * misgelopen winst + administratie	€ 870

Hieruit blijkt dat een retourzending met het afleveren van een nieuwe plaat economisch het meest voordelig is. Echter bestaat de kans ook dat de klant hier niet mee akkoord gaat en een terugstorting wilt, wat dan voor Decospan duurder uitkomt. Uit cijfers van Decospan blijkt dit slechts 10% van de gevallen voor te komen. Op basis hiervan kan er overgegaan worden naar een kostprijs berekening van een benadering met een manuele operator ten opzichte van een benadering met een AI-systeem. Hiervoor wordt gerekend, aan de hand van de vereiste, dat een AI-systeem 90% van de fouten kan detecteren (ten opzichte van het totaal aantal foutieve platen, aangezien het een operator vervangt). Tabel 4.26 vat de bijkomende kosten van foutdetectie door de klant samen. Deze cijfers komen overeen met financiële rapporten van Decospan, gemiddeld over 2022 en 2023.

Tabel 4.26 Vergelijking van kosten tussen manuele inspectie en AI-systeem.

Systeem	Platen bij klant	Kost door klant
Manuele inspectie	1352	€ 59 457
AI-systeem	463	€ 20 348
Verschil	889	€ 39 109

Ook dienen de vaste kosten per systeem bepaald te worden. Voor het manuele inspectie systeem kan de loonkost van een operator beschouwd worden. Echter plant Decospan om het AI-systeem parallel te laten lopen met de operator, waardoor de loonkost blijft. Tabel 4.27 vat de kost van een AI-systeem samen. Deze schatting werd gemaakt in het kader van het Tetra project door Captic.

Tabel 4.27 Kostenoverzicht van benodigde materialen AI-systeem. [24] [25]

Materiaal	**Prijs**
Camera	€925
Belichting	€700
Montage onderdelen	€450
Jetson	€680
Software	€49 500
Aanpassing machine lay-out en PLC programmatie	€12 000
Totaal	€64 255

In totaal is de aanschaffingsprijs van een AI-systeem rond de 64 duizend euro. Door het vermijden van extra kosten wegens klachten van klanten is er een jaarlijkse uitsparing van €39 109. Het rendement van de investering wordt bepaald aan de hand van een "IRR-berekening", afkomstig uit een cursus van Devolder et al. [26]. Deze start met de totale investeringskost, verwerkt vervolgens de jaarlijkse uitsparingen en houdt ook rekening met de jaarlijkse kosten voor onderhoud en elektriciteit, Deze worden op €3600 geschat. Ook worden de belastingen (34%) in rekening gebracht. De analyse wordt uitgevoerd over 5 jaar, op basis van de fiscale wetgeving. Machines moeten volgens de Belgische wet op 10 jaar afgeschreven worden en informatica op 3 jaar volgens Devolder et al. [26]. Als middenweg wordt 5 genomen. Figuur 4.32 geeft het resultaat van deze analyse weer.

Aankoop	64255	taks: 34%				
gebruik	5 jaar					
rendement	32,80%					
		jaar 1	jaar 2	jaar 3	jaar 4	jaar 5
besparingen		39.108,81	39.108,81	39.108,81	39.108,81	39.108,81
onderhoud		3.000,00	3.000,00	3.000,00	3.000,00	3.000,00
electr		600,00	600,00	600,00	600,00	600,00
afschrijvingen		12.851,00	12.851,00	12.851,00	12.851,00	12.851,00
netto voor belasting		22.657,81	22.657,81	22.657,81	22.657,81	22.657,81
tax		7.703,66	7.703,66	7.703,66	7.703,66	7.703,66
netto na belasting		14.954,15	14.954,15	14.954,15	14.954,15	14.954,15
cash flow		27.805,15	27.805,15	27.805,15	27.805,15	27.805,15
actuele waarde		20938,4007	15767,4617	11873,5357	8941,25205	6733,12402

Figuur 4.32: IRR berekening

Hieruit blijkt dat een investering in een AI-systeem een internal rate of return (IRR) heeft van 32,8% per jaar met een terugverdientijd van minder dan twee jaar. Dit ervan uitgaande dat de productie gelijk blijft en de inflatie niet verandert, wat in praktijk niet realistisch is. Niet alleen is dit een significante verbetering, die economisch interessant is, maar draagt dit ook bij aan de reputatie van Decospan. Indien een klant foutieve platen ontvangt, kan dit een negatief gevoel oproepen naar Decospan, wat slecht is voor de reputatie van het bedrijf. Het AI-systeem zorgt dat er 65.8% minder foutieve platen bij de klant terecht komen ten opzichte van een operator. Ook kan het potentieel nog meer fouten detecteren, want in praktijk dient niet iedere klant een klacht in bij een defect. Het implementeren van een AI-systeem ter kwaliteitscontrole kan dus gevolgen hebben voor een duurzaame groei van Decospan en gezien worden als een rendabele langetermijn investering.

Hoofdstuk 5

Algemeen besluit

5.1 State-of-the-art multi-class detectiemodel

Tijdens het onderzoek is er duidelijkheid gekomen over de onderzoeksvraag: "Kan de gewenste performantie bekomen worden met een "state-of-the-art" multi-class (supervised) detectiemodel, toegepast op alle data?". Deze vraag impliceert indirect dat er potentieel een öne-size-fits-alläpproach zou zijn. Typisch gebruikt de industrie hiervoor Yolo, omwille van de toegankelijke en geautomatiseerde API. Dit blijkt echter niet het geval te zijn voor Decospan omwille van twee redenen: enerzijds zijn de afbeeldingen van de Decospan dataset te groot, waardoor ze niet aan de meeste neurale netwerken gegeven kunnen worden. Anderzijds is er het probleem van class imbalance, waardoor er een bias ontstaat voor backgrounds. Door te testen op meerdere state-of-the-art modellen zoals PaDiM, Resnet en Yolo kan er besloten worden dat er voor de Decospan Dataset geen out of the box solution bestaat waar er zomaar afbeeldingen aan gevoed kunnen worden zonder een grondig doordachte pipeline op te stellen.

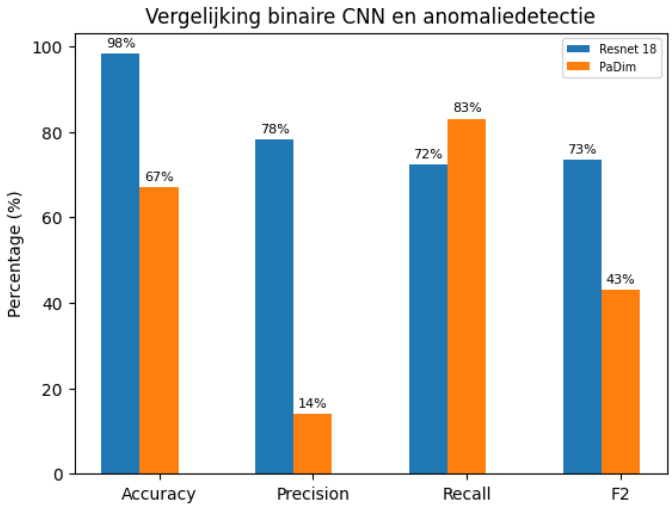
5.2 Unsupervised anomaliedetectie

Voor het deel anomaliedetectie werd de onderzoeksvraag "Kan een (unsupervised) anomaliedetectie model gebruikt worden om de defecten af te zonderen?önderzocht. Volgende methodologieën werden getest: K-means clustering en PaDiM. K-means clustering toont initieel veel potentieel, maar het behalen van een hoge nauwkeurigheid vormt nog steeds een substantiële uitdaging. Deze moeilijkheid is grotendeels toe te schrijven aan de experimentele natuur van de methode, waarbij het testen van een alternatieve scala aan backbones, lagen en dimensies soms tot een andere performantie kan leiden. Desondanks is dit proces aanzienlijk tijdrovend. Specifiek toegepast op de Decospan dataset, blijft de performantie laag. Dit is toe te schrijven aan de diverse eigenschappen, zoals houttype, patroon, kleur, esthetische imperfecties en randafwerkingen. De initiële fase van feature-extractie, gebruikmakend van

verschillende lagen, schiet tekort in het accuraat identificeren van deze kenmerken. Dit wijst mogelijk op de noodzaak van een geavanceerder model, dat bijvoorbeeld uit meerdere lagen en feature extractors bestaat en de mogelijkheid biedt om de gewichten van lagen aan te passen (training). Een meer geautomatiseerde variant hiervan zou PaDiM zijn. Deze zou op basis van patches een normale verdeling maken. Specifiek toegepast op de Decospan dataset bleek dit model niet effectief te zijn. De normale verdeling is te gevoelig voor ruis, die consistent aanwezig is in de Decospan omwille van specifieke variaties: houtkleur, houtsoort, specifieke patronen, randen, contrasten, etc. De enige oplossing om de performantie dichter bij de vereisten van Decospan te brengen was het trainen van een model per houtsoort, zonder randen. Dit verhoogt allereerst de engineering cost, maar ook de complexiteit qua integratie in de productielijn. Ook vraagt het om een apart model voor het verwerken van randen. Hierdoor is dit geen praktische oplossing voor de industrie. Uiteindelijk kan besloten worden dat anomaliedetectie veelbelovend oogt, maar met de huidige bestaande technieken niet robuust genoeg is voor het geautomatiseerd verwerken van real world data. Ofwel biedt de technologie geen ondersteuning voor ruis, ofwel vergt de technologie te veel manuele hyperparameter tuning. Beide opties maken het onbruikbaar in industriële toepassingen.

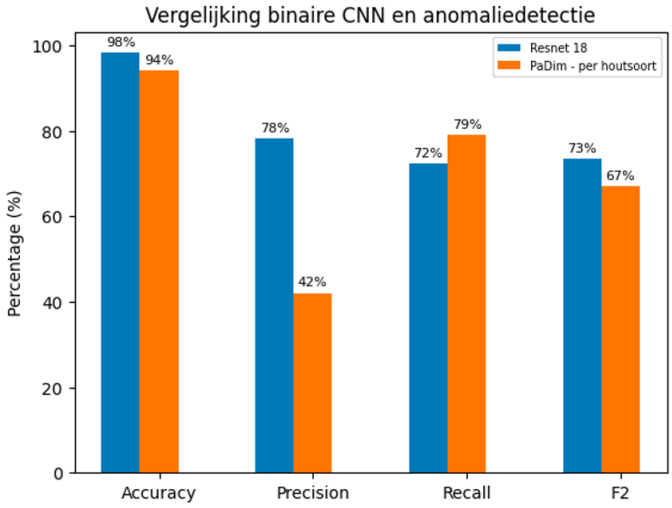
5.3 Binaire classificatie vs anomaliedectie

Dit hoofdstuk beantwoordt de onderzoeksvraag: "Kan een (supervised) binaire classifier gebruikt worden om de defecten af te zonderen en wat is de performantie tegenover anomaliedetectie?". Er wordt een vergelijking gemaakt tussen het beste model van de binaire classificatie: Resnet 18 en het beste anomalidetectiemodel: PaDiM met Resnet 18 backbone. Deze zijn beiden getraind op alle gesamplede backgrounds. Voor Resnet werden ook alle foutieve windows toegevoegd. Figuur 5.1 geeft hiervan het resultaat weer. Er is een groot contrast in performantie zichtbaar: de F2-score van Resnet ligt 30% hoger.



Figuur 5.1: Resnet18 vs PaDiM getraind op gesamplede dataset

Indien er gekeken wordt naar de maximaal mogelijke performantie uit de testen van hoofdstuk 4, staat Resnet 18 met de cut-out methode als binair classificatiemodel tegenover PaDiM als anomaliedetectiemodel. Hoewel ze niet getraind zijn op dezelfde dataset en hierdoor niet representatief vergeleken kunnen worden, geeft figuur 5.2 toch belangrijke inzichten. De F2 score van Resnet 18 ligt tot 5% hoger. Dit met als voordeel dat de houtsoort niet gekend hoeft te zijn en dat randen niet met een extra algoritme weggewerkt moeten worden, wat bij PaDiM wel het geval is. Dit maakt het gehele proces efficiënter in praktijk. Het nadeel hiervan is dat het binaire model supervised is en labels verwacht, wat in praktijk meestal een tijdsintensief manueel proces is.



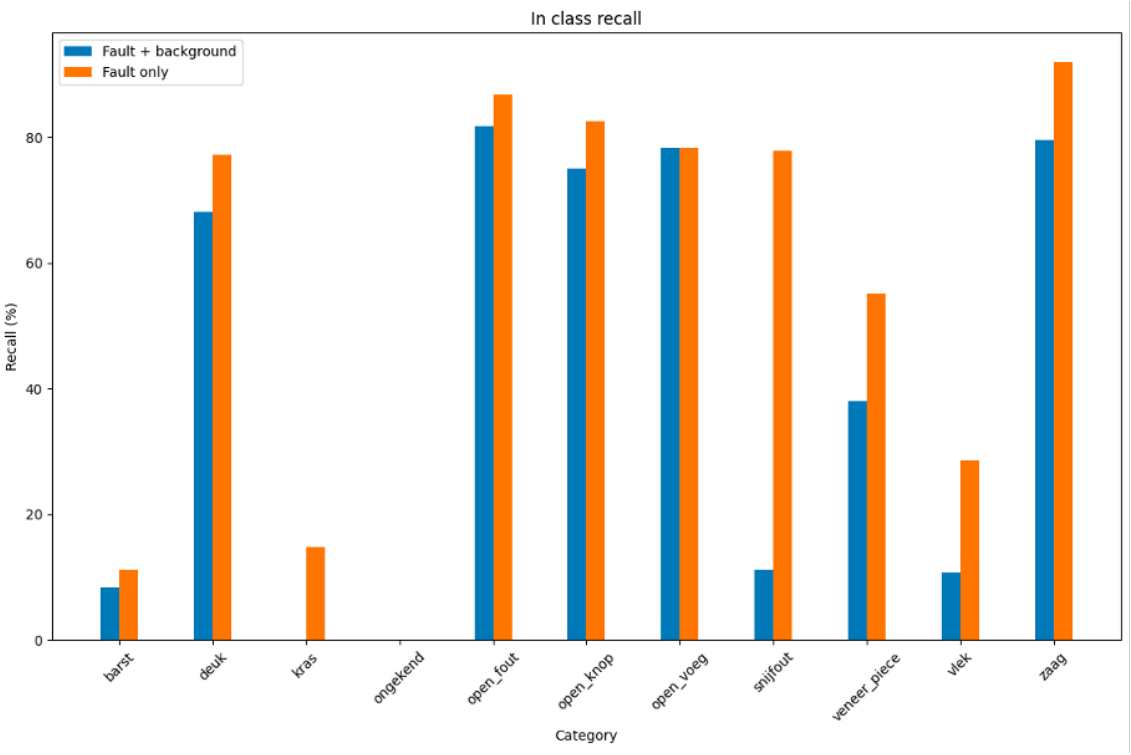
Figuur 5.2: Binary vs Anomaly detection

Hieruit kan besloten worden dat end-to-end deep learning modellen superieure performantie hebben en hierdoor toch de voorkeur ontvangen, ook al is er manuele annotatie nodig. Dit omwille van de beperkte complexiteit van de architectuur die de ontwerper dient samen te stellen. Echter blijkt de F2-score van anomaliedetectie te stijgen na een reeks optimalisaties (zoals bijvoorbeeld randen wegwerken of het trainen per houtsoort). Dit toont het potentieel voor toekomstige ontwikkelingen in anomaliedetectie. Indien deze erin slaagt hogere performantie te halen en robuust te zijn tegen ruis, zou de balans kunnen veranderen naar een voorkeur voor anomaliedetectie.

5.4 Multi-class CNN modellen vergeleken

Dit onderdeel vindt het antwoord op de onderzoeksvraag: "Kan een (supervised) multi-class classificatie model dat getraind is op enkel defecte data betere resultaten bieden dan een model getraind op alle data?". Grafiek 5.3 vergelijkt de in class recall van de 2 beste modellen van de multi-class CNN. Beiden getraind op een andere dataset: de ene op alles en de andere enkel op fouten. De accuracy tussen beiden vergelijken heeft geen zin, omdat het

eerste model meer dan 20 000 background windows meeneemt in deze metric. Vandaar de vergelijking in recall. Hieruit blijkt dat het "fault only" model in alle klassen beter presteert. Hieruit kan besloten worden dat door enkel op de foutklassen te focussen, het model sneller en efficiënter kan trainen. Het heeft minder categorieën om te leren, wat kan resulteren in een snellere convergentie naar een optimaal punt in de training.



Figuur 5.3: Vergelijking tussen multi-class CNN's getrained op verschillende data

5.5 Feature selection

Dit onderdeel omvat het selecteren van relevante features om de model performantie te verhogen. Enerzijds beatwoordt het de onderzoeksvraag: "Biedt feature selection een hogere model performantie voor de Decospan dataset?", maar beantwoordt het ook "Zorgt het trainen van een model per houtsoort voor een hogere performantie in vergelijking met een model dat geldt voor alle houtsoorten?". Allereerst werd er een directe vergelijking uitgevoerd voor het verschil in performantie te bekijken tussen binaire classifier getraind op alle houtsoorten en een binaire classifier getraind op een enkele houtsoort. Figuur 4.19 vat hiervan de resultaten samen. Hieruit blijkt dat de F2 score 1.9% hoger ligt. Mogelijks is dit een (beperkte) performantieverhoging of is dit de invloed van de beperkte dataset (split). Decospan heeft rond de 150 houtsoorten, waardoor er 150 modellen gemaakt, getraind en gefinetuned zouden moeten worden. Dit is praktisch niet haalbaar. Ook een verdere uitbreiding van de test naar een multi-class model werd overgeslaan, omwille van het beperkte aantal samples, dat

een onrealistische vergelijking zou opleveren.

Vervolgens werd een model getraind op de features die voor decospan het voornaamst zijn te detecteren. Hiervoor werden testen uitgevoerd op zowel het best presterende binaire model als best presterende multi-class model van de testen op alle input foutklassen. Hieruit bleek dat er geen noemenswaardige verbeteringen plaatsvonden en in het geval van de binaire classificatie de performantie er zelf op achteruit ging. Een mogelijke oorzaak hiervan is het verminderde aantal data, waardoor het modellen minder goed kan generaliseren. Dit valt het meest op bij de binaire classifier: die profiteert het meest van veel normale en abnormale sample windows.

5.6 Image preprocessing

Dit onderdeel beantwoordt het eerste deel van de onderzoeksvraag: "Welke methoden zijn effectief voor het voeden van een beperkt aantal hoge-resolutie afbeeldingen uit de Decospan dataset in een model, met inachtneming van real-time verwerkingsbeperkingen?".
Hiervoor worden er drie methoden voorgesteld die specifiek toepasbaar zijn op de probleemstelling van Decospan. Allereerst vindt het gegeven plaats dat de inputafbeeldingen een resolutie hebben. Dit levert twee problemen op: enerzijds ondersteunen neurale netwerken afbeeldingen met een beperkte resolutie bijvoorbeeld 224x224px, anderzijds moet aan het DORI-criterium, beschreven in hoofdstuk 2.2.3, voldaan zijn. Uit de literatuurstudie werd besloten dat de oplossing hiervoor het implementeren van het SAHI-algoritme is. Vervolgens kwam het imbalanced dataset probleem naar voren. Hiervoor werd de methodologie van sampling en augmentatie voorgesteld, om zo het aantal foutieve samples te balanceren met het aantal background samples. Tenslotte werd een derde methode voorgesteld om de performantie te verhogen: de "cut-out methode", besproken in hoofdstuk 3.5.1. Hierdoor worden alle fouten centraal geplaatst, waardoor er geen verkeerde ground truth-labels kunnen onstaan na het toepassen van het SAHI-algoritme.

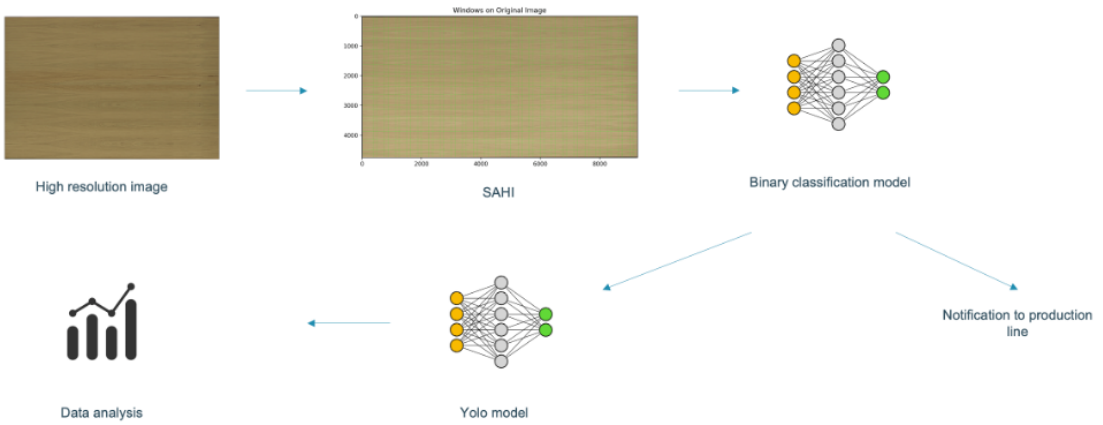
5.7 Voorgestelde oplossing

Voor de optimale pipeline voor deze probleemstelling, dient er een balans gemaakt te worden tussen enerzijds de performantie en anderzijds de inferentietijd. In een ideaal scenario is er een enkel model dat met hoge performantie aan multi-class classificatie doet, met eventueel segmentatie. In deze scriptie kwam het Yolo algoritme als veelbelovend naar voren om aan deze beschrijving te voldoen. Hierbij wordt duidelijk in hoofdstuk 4.4 dat de performantie van Yolo veelbelovend is, maar hoofdstuk 4.6 toont dat de inferentietijd te hoog ligt voor de probleemstelling van Decospan. Dit omwille van het SAHI-algoritme: volgens hoofdstuk 4.6 zijn er maximaal 924 windows per plaat die verwerkt moeten worden. Dit is niet realistisch met een time constraint van 3 seconden.

Uit de resultaten van 4.6 bleek dat het binair classificatiemodel met Resnet18 wel aan de constraint voldoet en bovendien een goede performantie heeft. Echter gaat de mogelijheid tot multi-class classificatie dan verloren. Daarom wordt er als aanbevolen oplossing gekeken naar de oplossingsmethode van Shi et al. [1], waar het "glance netwerk" wordt voorgesteld. Daar stellen ze voor een architecturaal eenvoudige binaire classifier te gebruiken, die enkel foute windows doorgeeft aan het 2e netwerk, om dan aan multi-class classificatie en segmentatie te doen. Dit met als doel de inferentietijd zo laag mogelijk te krijgen. Voor de Decospan probleemstelling kan deze 2 stage pipeline gebruikt worden in 2 mogelijke vormen: ofwel wordt deze methode gebruikt om binnen een redelijke tijd multi-class classificatie te hebben, niet-realtime, ofwel wordt deze methode gebruikt om multi-class classificatie te halen binnen de vereist tijd.

Aangezien de prioriteit van Decospan het detecteren van fouten is, en het optimaliseren van de investeringkost een belangrijke rol speelt, wordt de eerste methode voorgesteld. De fouten kunnen snel genoeg gedetecteerd worden, zodat de plaat van de productielijn verwijderd kan worden. Hoewel de predicties van het multi-class model dan niet realtime zijn, is er slechts enkele seconden vertraging bovenop de eerste alert. Voor een operator die de situatie monitort, is dit geen probleem. Het grote voordeel van deze methode is dat het compatibel is met een Jetson, waardoor deze oplossing kostenefficiënt is voor Decospan. Bovendien presteert een Yolo model dat enkel getraind is op foutieve windows beter dan ook op backgrounds, wat een win-win situatie is in dit geval: er is betere performantie en betere inferentietijd. Figuur 5.4 vat deze opstelling samen.

Als laatste werd een financiële analyse uitgevoerd: hieruit bleek dat dergerlijke installatie een terugverdientijd van 2 jaar heeft en vervolgens ook elk jaar een netto rendement van 32,8% oplevert. Naast het gegeven dat het financieel interresant is om deze technologie te gebruiken, vermindert het ook 65% van de foutieve platen bij klanten. Dit heeft een substantieel effect op de klantentevredenheid en reputatie van Decospan.



Figuur 5.4: Image processing pipeline

5.8 Future work

Allereerst kan besloten worden dat het verhogen van de performantie van een geïmplementeerd model essentieel is om aan de vereisten van Decospan te voldoen. Er moet gestreefd worden naar een recall van minstens 90% per foutcategorie. Indien de voorgestelde pipeline gebruikt wordt, wordt er eerst een binaire classifier gebruikt. Deze moet, bovenop de recall vereiste, meer dan 99% precision halen, zodanig de false positives beperkt blijven en er geen platen onterecht uit de productielijn gehaald worden. In de tweede stage kan er gestreefd worden naar het verderzetten van de 90% recall (per foutcategorie dan), maar dit is minder essentieel aangezien dit model enkel gebruikt wordt voor analyse en geen directe impact heeft op de initiële doelstelling: het detecteren van fouten in vlakke platen om deze uit de productielijn te verwijderen. In deep learning zijn er verschillende mogelijkheden om deze performantie te bereiken. Hoewel de literatuur spreekt over het vinden van een optimale architectuur (door middel van NAS) of hyperparameter tuning, blijken deze technieken enkel nuttig om de laatste percentages van een model te verbeteren. De oorzaak van de verlaagde performantie valt duidelijk te relateren aan het aantal samples. Hierdoor is de meest voor de hand liggend future work, het verzamelen van meer samples van iedere foutcategorie. Bij voorkeur zijn er van elke fout 500 instanties. Dit vormt een langdurig manueel arbeidsintensief proces, omwille van het detecteren van defecten en vervolgens het manueel annoteren. Hierdoor is er een opportuniteit om verder onderzoek te doen naar unsupervised learning, met name anomaliedetectie. Een syteem dat (met eventueel lagere performantie) unsupervised fouten uit nieuwe samples kan halen, biedt een grote tijdswinst. Hierdoor zou een operator enkel nog maar foute samples moeten labelen, in plaats van ze te zoeken. Omwille van de realtime constraint moet er met de huidige technologieën effectief gekozen worden voor een 2 stage approach. Er kan ook onderzoek gedaan worden naar alternatieve modellen die mogelijks met 1 stage een hoge accuracy en lage inferentietijd bevatten. Hierbij komen Vision transformer networks in aanmerking. Deze werden alreeds besproken in hoofdstuk 2.2.2.2. Omwille van de grote hoeveelheden data vereist om het netwerk te trainen, zou er beroep gedaan moeten worden op pretrained modellen. Omwille van de recentheid van deze technologie, blijft het grotendeels beperkt tot conceptuele literatuur en zijn er weinig praktische implementaties.

Concreet kunnen volgende stappen aangeraden worden aan Decospan om deze probleemstelling op te lossen:

1. Het verzamelen van meer samples. Aangezien de operator nu al jaarlijks 3275 platen detecteert en de lijnscan-camera alreeds geïmplementeerd is, vergt het vrij weinig moeite om een extensie aan het huidige systeem te programmeeren dat detecteert wanneer een operator een plaat afkeurt. Zo wordt de huidige dataset met alreeds 371 afbeeldingen vertienvoudigd. Ruwe annotatie met bounding boxes wordt geschat op 30 seconden per afbeeldingen, terwijl annotatie met masks geschat wordt op 3 à 4 minuten per afbeelding. Een expert van Decospan zou de ruwe annotatie op zich kunnen nemen, terwijl geautomatiseerde software of derden het eenvoudige, maar tijdsintensieve werk

doen van masks tekenen.

2. Het opnieuw trainen van de modellen in de voorgestelde pipeline met de cut-out methode. Hier zou een performantie verhoging waarneembaar moeten zijn.

3. Het implementeren in de productielijn en het programmeren van een visuele weergave voor de operator.

4. Het opslaan van resultaten in een algemene databank voor verdere data-analyse.

Bibliografie

[1] J. Shi, Z. Li, T. Zhu, D. Wang en C. Ni, "Defect Detection of Industry Wood Veneer Based on NAS and Multi-Channel Mask R-CNN", *Sensors*, jrg. 20, nr. 16, 2020, issn: 1424-8220. doi: 10.3390/s20164398. adres: https://www.mdpi.com/1424-8220/20/16/4398.

[2] R. Cohn en E. Holm, "Unsupervised Machine Learning Via Transfer Learning and k-Means Clustering to Classify Materials Image Data", *Integrating Materials and Manufacturing Innovation*, jrg. 10, nr. 2, p. 231–244, apr 2021, issn: 2193-9772. doi: 10.1007/s40192-021-00205-8. adres: http://dx.doi.org/10.1007/s40192-021-00205-8.

[3] T. Defard, A. Setkov, A. Loesch en R. Audigier, "PaDiM: A Patch Distribution Modeling Framework for Anomaly Detection and Localization", in *Pattern Recognition. ICPR International Workshops and Challenges*, A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante en R. Vezzani, red., Cham: Springer International Publishing, 2021, p. 475–489.

[4] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger en C. Steger, "The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection", *International Journal of Computer Vision*, jrg. 129, nr. 4, p. 1038–1059, 2021. doi: 10.1007/s11263-020-01400-4.

[5] R. B. Girshick, J. Donahue, T. Darrell en J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", *CoRR*, jrg. abs/1311.2524, 2013. arXiv: 1311.2524. adres: http://arxiv.org/abs/1311.2524.

[6] M. Gurucharan, *Basic CNN architecture: Explaining 5 layers of Convolutional Neural Network — upgrad blog*, jul 2022. adres: https://www.upgrad.com/blog/basic-cnn-architecture/.

[7] K. Yang, Y. Wang, S. Fan en A. Mosleh, "Multi-Criteria Spare Parts Classification Using the Deep Convolutional Neural Network Method", *Applied Sciences*, jrg. 11, p. 7088, jul 2021. doi: 10.3390/app11157088.

[8] F. C. Akyon, S. Onur Altinuc en A. Temizel, "Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection", in *2022 IEEE International Conference on Image Processing (ICIP)*, IEEE, okt 2022. doi: 10.1109/icip46576.2022.9897990. adres: http://dx.doi.org/10.1109/ICIP46576.2022.9897990.

[9] C. Li, B. Zhang, H. Hu en J. Dai, "Enhanced Bird Detection from Low-Resolution Aerial Image Using Deep Neural Networks", *Neural Processing Letters*, jrg. 49, jun 2019. doi: 10.1007/s11063-018-9871-z.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit en N. Houlsby, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, 2021. arXiv: 2010.11929 [cs.CV].

[11] K. Bärudde en M. Ganda, *Industrial 3D anomaly detection and localization using ...* 2023. adres: https://liu.diva-portal.org/smash/get/diva2:1766718/FULLTEXT01.pdf.

[12] Y. Ge, D. Jiang en L. Sun, "Wood Veneer Defect Detection Based on Multiscale DETR with Position Encoder Net", *Sensors*, jrg. 23, nr. 10, 2023, issn: 1424-8220. doi: 10.3390/s23104837. adres: https://www.mdpi.com/1424-8220/23/10/4837.

[13] M. Gao, D. Qi, H. Mu en J. Chen, "A Transfer Residual Neural Network Based on ResNet-34 for Detection of Wood Knot Defects", *Forests*, jrg. 12, nr. 2, 2021, issn: 1999-4907. doi: 10.3390/f12020212. adres: https://www.mdpi.com/1999-4907/12/2/212.

[14] M. Mohsin, O. Balogun, K. Haataja en P. Toivanen, "Convolutional neural networks for real-time wood plank detection and defect segmentation", *F1000Research*, jrg. 12, p. 319, 2023, version 1; peer review: 1 approved with reservations. doi: 10.12688/f1000research.131905.1.

[15] A. Urbonas, V. Raudonis, R. Maskeliūnas en R. Damaševičius, "Automated Identification of Wood Veneer Surface Defects Using Faster Region-Based Convolutional Neural Network with Data Augmentation and Transfer Learning", *Applied Sciences*, jrg. 9, nr. 22, 2019, issn: 2076-3417. doi: 10.3390/app9224898. adres: https://www.mdpi.com/2076-3417/9/22/4898.

[16] A. Benali Amjoud en M. Amrouch, "Convolutional Neural Networks Backbones for Object Detection", in *Image and Signal Processing*, deel 12119, Springer, 5 jun 2020, p. 282–289. doi: 10.1007/978-3-030-51935-3_30.

[17] H. Zhang, C. Hao, W. Song, B. Jiang en B. Li, "Adaptive Slicing-Aided Hyper Inference for Small Object Detection in High-Resolution Remote Sensing Images", *Remote Sensing*, jrg. 15, nr. 5, 2023, issn: 2072-4292. doi: 10.3390/rs15051249. adres: https://www.mdpi.com/2072-4292/15/5/1249.

[18] P. Durai, *Exploring Sahi: Slicing aided hyper inference for small object detection*, dec 2023. adres: https://learnopencv.com/slicing-aided-hyper-inference/.

[19] M. Gartz, *How to detect small objects in (very) large images*, mei 2023. adres: https://blog.ml6.eu/how-to-detect-small-objects-in-very-large-images-70234bab0f98.

[20] Marcinrutecki, *Best techniques and metrics for Imbalanced Dataset*, dec 2022. adres: https://www.kaggle.com/code/marcinrutecki/best-techniques-and-metrics-for-imbalanced-dataset.

[21] Labelmeai, *Labelmeai/labelme: Image polygonal annotation with python (polygon, rectangle, circle, line, point and image-level flag annotation)*. adres: https://github.com/labelmeai/labelme.

[22] Openvinotoolkit, *Openvinotoolkit/anomalib: An anomaly detection library comprising state-of-the-art algorithms and features such as experiment management, hyper-parameter optimization, and edge inference.* adres: https://github.com/openvinotoolkit/anomalib.

[23] X. Jiang, J. Liu, J. Wang, Q. Nie, K. WU, Y. Liu, C. Wang en F. Zheng, "SoftPatch: Unsupervised Anomaly Detection with Noisy Data", in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho en A. Oh, red., deel 35, Curran Associates, Inc., 2022, p. 15 433–15 445. adres: https://proceedings.neurips.cc/paper_files/paper/2022/file/637a456d89289769ac1ab29617ef7213-Paper-Conference.pdf.

[24] T. Desmet, *Dataset Decospan*, E-mail communication, Received by Karel Debedts, mei 2024.

[25] M. D. Ryck, *Info financiële aspect Captic*, E-mail communication, Received by Karel Debedts, apr 2024.

[26] K. Devolder, *Ingenieur en economie*, Course taught at KU Leuven, KU Leuven, Faculty of Engineering, 2024.