KU LEUVEN

FACULTEIT INDUSTRIËLE
INGENIEURSWETENSCHAPPEN

CAMPUS BRUGGE

Inline lengtebepaling van vissen per soort

Viktor LAMBERT

Promotor: Dr. ing. M. De Ryck

Co-promotor: S. Delacauw

Masterproef ingediend tot het behalen van de graad van master of Science in de industriële wetenschappen: Elektromechanica

Academiejaar 2022 - 2023

KU LEUVEN

FACULTEIT INDUSTRIËLE
INGENIEURSWETENSCHAPPEN

CAMPUS BRUGGE

Inline lengtebepaling van vissen per soort

Viktor LAMBERT

Promotor: Dr. ing. M. De Ryck

Co-promotor: S. Delacauw

Masterproef ingediend tot het behalen van de graad van master of Science in de industriële wetenschappen: Elektromechanica

Academiejaar 2022 - 2023

Inhoud

Figuurlijst

1 Introductie

1.1 Context

Deze thesis werd uitgevoerd bij ILVO. ILVO is een onafhankelijk wetenschappelijk onderzoekscentrum van de Vlaamse overheid. De naam ILVO staat voor het Instituut voor Landbouw-, Visserij- en Voedingsonderzoek. Ze krijgen de taak van de overheid om mee te werken aan de verduurzaming van deze sectoren. Dit gebeurt in Vlaanderen, maar ook in België en Europa. [8] In dit project is het de bedoeling dat er een beter zicht gekregen wordt op de lengteverdeling van de vissen. Dit heeft een ecologisch doel in contrast tot een economisch doel. De implementatie zal eerst op de visveiling gebeuren. Hier worden de vissen verenkeld op een lopende band geplaatst. Vervolgens worden er beelden vastgelegd. Deze beelden worden geanalyseerd. De data wordt dan gebruikt om een verdelingsplot op te stellen. Op termijn zal het systeem ook aanboord de vismijn geplaatst worden. Hier zijn er echter meer storingsfactoren zoals overlappende vissen.

1.2 Hoofddoelen

Het hoofddoel van het project is om een spreiding te krijgen van de lengte van de gevangen vissen per soort. Deze taak kan opgedeeld worden in 2 onderdelen. In een eerste onderdeel zal de vis worden gedetecteerd en de soort worden bepaald. Het doel is om hier een algoritme te ontwikkelen en valideren die in een eerste stap een vis kan detecteren in de afbeelding en vervolgens de soort kan bepalen. In een tweede onderdeel zal er een algoritme ontwikkeld en gevalideerd worden die de lengte van de vis bepaalt.

1.3 Subdoelen

1.3.1 Detectie en classificatie

Verkregen data verwerken

Het neuraal netwerk die de detectie zal maken zal eerst moeten getraind worden. Dit zal gebeuren door een grote dataset te maken met gelabelde foto's. ILVO heeft 2 datastromen voorzien om in de dataset op te nemen. Enerzijds zijn er een aantal video's opgenomen in een gecontroleerde setting. Anderzijds werd de inspanning geleverd door ILVO om handmatig een aantal afbeeldingen te labelen door middel van Labelbox.

Algoritmen selecteren en trainen

Nadat de data verwerkt is, kan er geselecteerd worden welke soorten algoritmen en met welke parameters er getraind worden. Hierbij zullen er meerdere variaties getraind worden. De uitvoer van dit algoritme zal een label met de voorspelde soort en een bounding box met de voorspelde locatie zijn.

Algoritmen valideren

Om de verschillende variaties te vergelijken zullen deze gevalideerd worden door middel van een steekproef. De score hierop zal dus bepalen hoe performant en consistent het algoritme is.

1.3.2 Lengte meting

Ontwikkeling algoritme

Er zullen een aantal verschillende algoritmes ontwikkeld worden om de lengte zo juist mogelijk op te meten. Deze algoritmes zullen verder bouwen op het eerste algoritme. Hier zullen ze dus de bounding box als hulp gebruiken. De uitvoer van deze algoritmes zal een afstand in pixel coördinaten zijn. Deze zullen nog moeten omgezet worden naar wereld coördinaten om een bruikbaar resultaat in centimeter te produceren. Verdere uitleg hierover zal volgen in het desbetreffende hoofdstuk.

Validatie algoritme

Er is geen data beschikbaar over de lengte van de vissen die te zien zijn op de trainingsdata. Hierdoor kan er geen spreiding berekend worden. Om toch een beeld te krijgen van de performantie van het algoritme zal de ruggengraat detectie manueel worden gecontroleerd.

1.4 Structuur van de scriptie

In hoofdstuk 2 zal de literatuurstudie besproken worden. Hierbij wordt eerst uitleg gegeven over de verschillende soorten van object detectie. Ook zullen gelijkaardige bestaande systemen besproken

worden. Op basis van de literatuurstudie zullen beslissingen genomen worden over de gebruikte technieken.

In hoofdstuk 3 wordt er meer uitleg gegeven over de gebruikte methodes om de hoofd- en subdoelen te bereiken. Eerst zal er verduidelijkt worden hoe de dataset is samengesteld. Vervolgens zal er uitleg gegeven worden over welke netwerken er zullen getraind worden en met welke parameters. Ook zal de gebruikte software en hardware besproken worden. Ten slotte wordt er uitleg gegeven over verschillende algoritmen om de lengte op te meten van de gedetecteerde vis.

In hoofdstuk 4 worden de validatie technieken verduidelijkt. Ook worden de resultaten besproken. Er zal een zicht gekregen worden op de performantie en consistentie van de algoritmen. Ook zal de proces tijd gemeten en besproken worden.

In hoofdstuk 5 zullen de conclusies genomen worden uit de resultaten. Ook zal er besproken worden wat er nog future work is en waar er nog op kan worden verbeterd.

2 Literatuurstudie

2.1 Detectie en classificatie

2.1.1 Inleiding

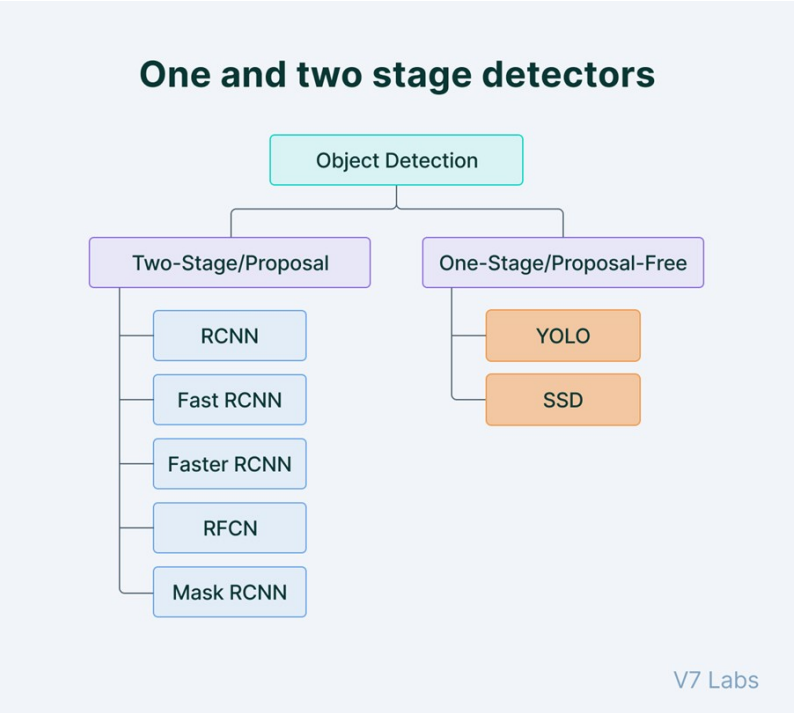
In dit project zal er gebruik gemaakt worden van een convolutioneel neuraal netwerk. Dit is een techniek die bij zeer veel toepassingen gebruikt wordt zoals object detectie, maar ook tekst en foto generatie en nog vele andere. Eerst wordt CNN zelf en variaties op CNN verduidelijkt. Vervolgens wordt er gekeken naar bestaande technieken om deze technologie te gebruiken bij het detecteren en classificeren van vissoorten.

2.1.2 Object detectie

CNN

Een CNN of convolutioneel neuraal netwerk is een algemene term voor een neuraal netwerk die convolutie gebruikt. Een CNN wordt vaak gebruikt bij foto's of andere data waar de spatiële informatie van de pixels belangrijk is. Dit wordt behaald door het gebruik van convolutielagen. Deze lagen werken op het volgende principe: er wordt een kernel gebruikt die over de afbeelding glijdt en het resultaat zal opslaan in de volgende laag. De volgende laag zal dus een convolutie zijn van de huidige laag. De kernels gedragen zich als filters waardoor bepaalde features op de afbeelding worden herkend. Deze features kunnen simpele lijnen of hoeken zijn, maar ook complexere zaken zoals ogen of staarten.[9] Er zijn veel variaties op deze methode die elk hun eigen voor- en nadelen hebben. De variaties kunnen worden opgesplitst in twee categorieën zoals te zien in Figuur 2.1:de Two-stage algoritmen en de One-stage algoritmen. Het verschil tussen deze categorieën is dat er bij Two-stage een voorbereiding nodig is om vervolgens het CNN te laten werken. De voorbereiding is een algoritme op zich die de afbeelding bijvoorbeeld opdeelt in verschillende regio's die dan elk apart verwerkt worden door het CNN. Bij One-stage wordt het CNN direct toegepast op de originele afbeelding. [1] Verder kunnen netwerken verschillen in de architectuur zelf. Naast een input en output layer kunnen netwerken gebruik maken van pooling

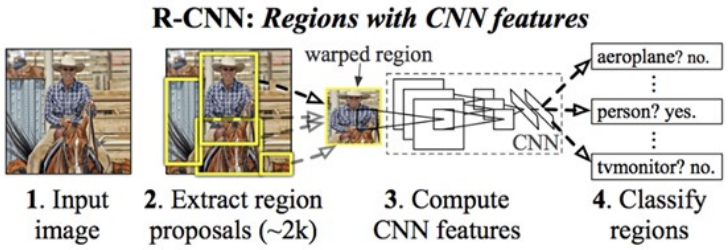
layers, flattening layers of fully connected layers. Deze layers verwerken de interne data op een bepaalde manier die gunstig kan zijn voor het gewenste doel.



Figuur 2.1: Soorten Object Detectie [1]

RCNN

De R in RCNN staat voor ”Region”. Dit is een subcategorie van CNN’s. Een RCNN is een voorbeeld van een Two-stage algoritme. Hierbij wordt een beeld eerst ingedeeld in veel mogelijk interessante regio’s. Dit gebeurt door middel van een algoritme die gelijkaardige regio’s samen neemt. De naam van dit algoritme is ”selective search”. Vervolgens worden deze regio’s elk apart gegeven aan een CNN die hierop een classificatie uitvoert, zoals te zien in Figuur 2.2.



Figuur 2.2: Werking RCNN [2]

Een nadeel aan dit algoritme is dat deze methode niet real-time kan uitgevoerd worden. Ook worden de regio’s gekozen door een algoritme dat niet leert, aangezien dit een statisch algoritme is dat niet verandert. Daarbij is dit algoritme vrij traag. Een variatie op RCNN is Faster RCNN.

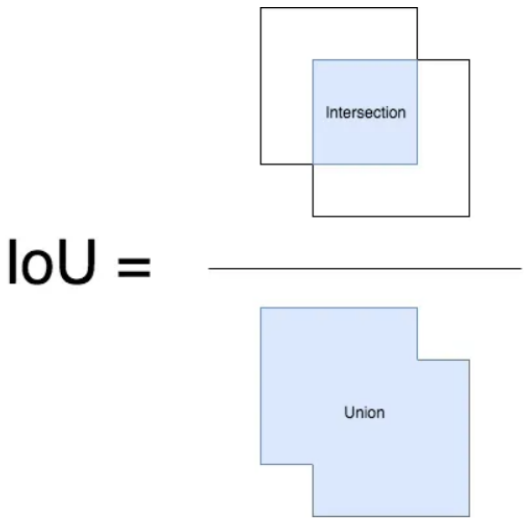
Deze methode is gelijkaardig aan RCNN, maar hier is het algoritme dat de regio's selecteert een neuraal netwerk in plaats van een statisch algoritme. Dit heeft als gevolg dat deze sneller en beter kan worden. [2]

YOLO

You Only Look Once, afgekort als YOLO, is een One-stage algoritme. Deze deelt het scherm eerst in, in meerdere grids van verschillende celbreedtes. In elk van deze cellen wordt er een CNN gebruikt. Deze zal dus per cel een voorspelling doen van wat er te zien is. Dit zal ervoor zorgen dat hetzelfde object meermaals wordt gedetecteerd. Om dit tegen te gaan wordt er ten slotte een Non Maximal Surpression toegepast. Deze zorgt ervoor dat er per object 1 bounding box en label verkregen wordt, zoals te zien op Figuur 2.3.

Non Maximal Surpression of NMS is een algoritme die gebruikt wordt om uit een groot aantal voorgestelde bounding boxes, een kleiner aantal zekere bounding boxes te kiezen. Eerst wordt de bounding box met de hoogste zekerheid geselecteerd. In stap 2 zullen de intersection over union of IOU berekend worden voor elke andere mogelijke bounding box. Als deze IOU groter is dan een bepaalde threshold, wordt deze bounding box verworpen. Als dit voltooid is, wordt opnieuw de bounding box met de hoogste zekerheid geselecteerd. Hier worden opnieuw alle IOU berekend en overlappende bounding boxes verworpen. Dit proces herhaalt zich tot alle resterende bounding boxes geselecteerd zijn.

Intersection over Union of IOU is een techniek die bepaalt hoeveel twee rechthoeken overlappen. Eerst wordt de intersection of doorsnede berekend. Deze wordt gedeeld door de union of unie. De IOU van twee rechthoeken die niet overlappen is nul en de IOU van tweemaal dezelfde rechthoek is één.



Figuur 2.3: Visualisatie IOU [3]

Een groot voordeel aan YOLO is dat deze in real-time kan werken. Dit algoritme heeft ook enkele nadelen. YOLO heeft het moeilijk met de detectie van kleine objecten in een groep. Dit is zo omdat er per cel maar 1 detectie gebeurt. Als een object dus kleiner is dan een cel, zal deze maar 1 keer gedetecteerd worden.

In YOLO v2 werd dit probleem opgelost door de cellen nog verder op te delen. Vervolgens werden er in YOLO 9000 veel meer klassen toegevoegd. YOLO v3 heeft een betere nauwkeurigheid, maar is een stuk trager. Verdere variaties van YOLO hebben minder duidelijke voordelen. Hierover wordt er meer uitleg gegeven in de documentatie over de verschillende versies. Ook zal YOLO een lagere accuracy hebben tegenover een trager algoritme zoals RCNN. [1]

Retinanet

Retinanet is een One-Stage algoritme. Hier wordt er gewerkt met een Focal Loss functie. Dit zorgt ervoor dat er meer nadruk wordt gelegd op de moeilijke negatieve trainingsdata. Dit zorgt ervoor dat de makkelijke voorbeelden minder gebruikt worden tijdens het trainen. Dit betekent dat het neuraal netwerk beter zal leren en dus sneller, betere resultaten zal bereiken. [10]

Resnet

Resnet is een CNN model waarbij er gebruik gemaakt wordt van residual learning. In de paper over resnet wordt er empirisch bewijs geleverd die zegt dat deze netwerken makkelijker te optimaliseren zijn en betere nauwkeurigheid verkrijgen. Resnet is een voorbeeld van transfer learning. Hierbij wordt een netwerk niet geïnitialiseerd op willekeurige waarden, maar worden de middelste lagen van een reeds getraind netwerk gebruikt. Hierdoor kan er sneller getraind worden, omdat de basisfeatures al vooraf aangeleerd zijn. Resnet is ook geïntegreerd in pytorch waardoor het makkelijk is om dit model te gebruiken. [11]

2.1.3 Bestaande systemen

Dit project is zeker niet de eerste poging om een detectie algoritme te maken voor vissen. Er zijn al een aantal papers geschreven die een zeer aansluitend doel hadden en dus nuttig zijn om te verwerken in de scriptie. We kunnen hierbij de datasets en de nauwkeurigheid vergelijken. Ook wordt er vermeld welk CNN model er wordt gebruikt.

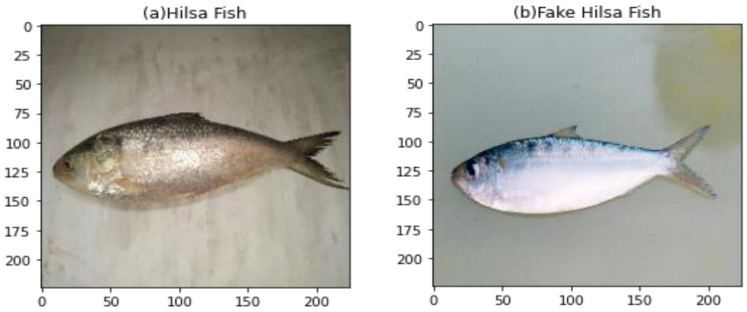
Visclassificatie met Pytorch resnet

Een zeer gelijkaardig project werd gevonden op de website Blacksuan19. Hier wordt een neuraal netwerk ontwikkeld en getraind die 9 verschillende vissoorten kan onderscheiden. Er wordt gebruik gemaakt van resnet. De versie van resnet is niet vermeld in deze literatuur. De dataset bestaat uit 2000 foto's per vissoort. De dataset bestond uit zowel gewone afbeeldingen, als masks van een bepaalde vis. Deze vissen waren telkens gecentreerd en er was een duidelijk contrast met de blauwe achtergrond. Dit zorgt voor een grote en duidelijke dataset. Data augmentatie werd ook toegepast door de afbeeldingen te draaien rond het middelpunt. Aangezien alle vissen gecentreerd waren, zorgde dit voor even bruikbare en representatieve afbeeldingen. Er werd getraind voor 10 epochs, wat vrij weinig lijkt. Als de training loss grafiek bekeken werd kon er gezien worden dat deze nog potentieel heeft om te dalen. Het is dus mogelijk dat er met meer epochs te trainen, betere resultaten kunnen worden behaald. Ook kan de keuze om masks te verwerken in de trainingsdata in twijfel getrokken worden. Er zit andere informatie in een mask die niet overeen komt met de informatie van gewone afbeeldingen. Het is dus ook mogelijk dat er een beter resultaat bekomen wordt als deze masks niet gebruikt worden in de trainingsdata. Er werd een nauwkeurigheid van

87% behaald. Hiermee wordt bedoeld dat 87% van de vissen juist geclassificeerd werden door het model.[12]

Hilsa vis detectie

M. Islam, J. F. Ani, A. Rahman, en Z. Zaman schreven een paper waarin er beschreven werd hoe er een neuraal netwerk gebruikt werd om het onderscheid te kunnen maken tussen Hilsa vissen en zeer gelijkaardige ”Fake Hilsa” vissen. Beide zijn te zien op Figuur 2.4. Deze paper leek relevant aangezien sommige soorten vissen waartussen er onderscheid moet gemaakt worden in dit project, ook vrij gelijkaardig zijn. De gebruikte dataset bestond uit 16,622 afbeeldingen. Deze werd verder opgedeeld in ongeveer de helft echte Hilsa afbeeldingen en de helft ”Fake Hilsa” vissen. De ”Fake Hilsa” vissen waren eigenlijk verschillende soorten sardienen en andere gelijkaardige soorten. Ook werden alle afbeeldingen geschaald naar 224 x 224. Er werden vijf verschillende soorten CNN netwerken gebruikt. Namelijk: NASNetMobile, VGG16, Xception, InceptionV3, DenseNet201. Elk netwerk kon trainen voor 40 epochs. Hieruit bleek dat DenseNet201 de beste resultaten gaf met een nauwkeurigheid van 97.02%. NASNetMobile, VGG16, Xception en InceptionV3 behaalden een nauwkeurigheid van 86.75%, 94.61%, 96.60% en 96.87% respectievelijk. [4]



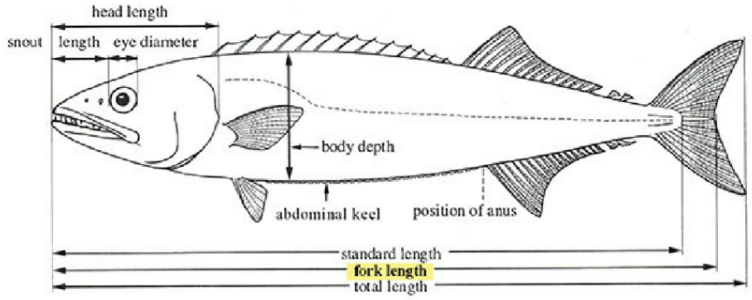
Figuur 2.4: Afbeeldingen van de Hilsa vis links en de ”fake Hilsa vis” rechts [4]

2.2 Lengte bepaling van vis

2.2.1 Inleiding

In het tweede deel van het project zal de lengte gemeten worden van de vis. Dit kan op veel verschillende manieren gebeuren. Het is belangrijk om een vis op te meten omdat hier veel informatie kan uit gehaald worden. Er kan bijvoorbeeld een beeld gekregen worden op soort groei of krimp. Ook moeten vissers de lengte van hun vangst bijhouden zodat deze conform zijn met de regulering in hun regio.

De lengte van een vis is op zich een ambigue term. Er zijn namelijk veel lengtematen die een vis karakteriseren zoals te zien in Figuur 2.5. In deze thesis wordt de totale lengte opgemeten. Deze wordt gedefinieerd als de lengte van het meest vooruitstekende deel van de snuit tot het achterste deel van de staart. Hierbij is de mond gesloten en de staartvin ontspannen. [13]



Figuur 2.5: Verschillende lengtematen op een vis [5]

2.2.2 Bestaande systemen

Automated measurement of species and length of fish by computer vision

In 2006 is er een paper geschreven waarin praktisch dezelfde case werd uitgewerkt als in deze scriptie. Er werden 7 vissoorten onderscheid van elkaar met een 99,8% nauwkeurigheid. Dit betekent dat 99,8% van de afbeeldingen een juiste classificatie kregen. Ook werd de lengte gemeten van elke vis met een standaardafwijking van 1,2 mm. Deze resultaten werd echter niet behaald via machine learning. Er werd een statisch programma geschreven die de detectie maakt. Klassieke machine visie technieken werden gebruikt zoals edge detection en vergelijkingen van dikte per meetpunt. Deze methode blijkt zeer goed te werken, maar is onderhevig aan fouten als er een nieuwe soort moet gedetecteerd worden. De volledige code zou opnieuw moeten geschreven worden.[14]

The Measurement of Fish Size by Machine Vision - A Review

In 2017 werd een paper geschreven die enkele methodes samenvat en vergelijkt om vissen te meten met machine visie. Hierbij werd een methode met Hough transformations vergeleken met een gebogen lijn die door de middelpunten van de vis ging. Hierbij werd duidelijk dat de gebogen lijn een lagere fout had. Dit was 5% en 1% respectievelijk. Deze paper leert ons dat een meting conform met de kromming van de vis de beste resultaten geeft. [15]

3 Methodologie

3.1 Data captatie

3.1.1 Oorsprong

Er waren 2 verschillende datastromen ter beschikking.

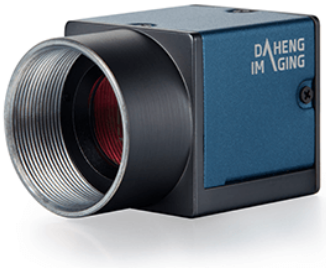
IVLO heeft een aantal video's ter beschikking gesteld om de dataset te creëren. In totaal zijn er 18 video's van ongeveer 5 tot 10 minuten lang, verdeeld over 4 vissoorten. Deze soorten zijn pladijs, schar, tong en tongschar.

In het tweede semester heeft ILVO de dataset uitgebreid tot 10 vissoorten. Deze kwam in de vorm van een Labelbox render. Dit wilt zeggen dat er handmatig een bounding box en soort aangeduid werd door middel van de software Labelbox. Deze software is gebruiksvriendelijk en laat toe om een groot aantal afbeeldingen van een label te voorzien.

3.1.2 Camera en lens

Op locatie zijn er 2 identieke camera's gemonteerd. In dit project wordt hier maar 1 van gebruikt. Het type van de camera is MER2-302-56U3C, te zien op Figuur 3.1. Deze kan verbonden worden met een pc door middel van een USB 3.0 connectie. Hierdoor is deze dus makkelijk te gebruiken. Ook heeft deze camera een C-mount, wat een populair type verbinding is om een lens te bevestigen. Hij heeft een resolutie van 2048x1536 met een pixelgrootte van 3.45µm. Ook kan hij tegen schommelende temperaturen en humiditeiten van 0℃ tot 45℃ en 10% tot 80% respectievelijk. [16]

De gebruikte lens heeft al type LCM-5MP-04MM-F2.0-1.8-ND1, te zien in Figuur 3.2. Deze kan gemonteerd worden doormiddel van de C-mount. De focale afstand is 4mm en de apature is F2.0. [7]



Figuur 3.1: MER2-302-56U3C camera [6]

Figuur 3.2: LCM-5MP-04MM-F2.0-1.8-ND1 lens [7]

3.1.3 Belichting

In deze case werd er geen extra belichting toegevoegd. Enkel de reeds aanwezige tl-lampen geven licht aan de afbeeldingen. Dit gaf tijdens de testen geen nadelige effecten op de vastgelegde beelden. Ook het kleur was een neutraal wit, die de kleur van de vissen correct weergaf.

3.2 Dataset creatie

3.2.1 Doel

Om het neurale netwerk te trainen zal er een dataset nodig zijn. De structuur van deze dataset is een foto met een bijhorend label. Het label hierbij is de soort vis. Deze soort wordt weergegeven door middel van de fao code van de soort. Hierbij kan elke soort weergegeven worden met een code van 3 letters. Bijvoorbeeld: PLE staat voor pladijs, GUU staat voor rode poon,... De beste resultaten worden behaald als op de foto's 1 vis te zien is met een representatieve achtergrond.

3.2.2 Dataset video

Foto's uit video's

De eerste stap in de creatie van de dataset is het omzetten van de video's naar foto's. Er wordt gebruik gemaakt van de library PyAv om deze conversie te maken. Om tijd uit te sparen en te grote mappen te vermijden, zal er om de 10 frames een beeld opgeslagen worden. Dit resulteerde in ongeveer 8 beelden per individuele vis. Deze werden ook gecropt zodat enkel de lopende band afgebeeld staat, zoals te zien op de eerste foto op Figuur 3.3. Dit werd gedaan voor elke video. Er moet eenmaal per video vermeld worden in de software waar de lopende band zich bevindt in de video. Deze coördinaten kunnen worden gevonden door een enkele frame in Paint in te laden. Vervolgens wordt de cursor geplaatst in de linkerbovenhoek van de lopende band. Hier kunnen de coördinaten afgelezen worden. Ten slotte kan de cursor in de rechteronderhoek van de lopende band geplaatst worden. Hier kunnen opnieuw de coördinaten afgelezen worden.

Verschil

In de eerste methode werd het verschil genomen tussen de huidige frame en een achtergrond frame. Dit gaf een vaag beeld van de vis als resultaat. Het nadeel van deze methode is dat de vis niet het enige is die verandert in de frame. De lopende band zelf is niet volledig egaal en wordt dus ook gedetecteerd als een verschil. Een groot probleem trad op, op frames waar de belichting werd weerspiegeld op de natte lopende band. Dit gaf een reflectie die elke frame anders was, waardoor deze telkens een duidelijk verschil gaf. Ook werd de vis vaak niet volledig gedetecteerd, waardoor deze niet volledig werd afgebeeld in het resultaat. Na een periode experimenteren werd de keuze gemaakt om op een andere methode over te schakelen.

Chroma-keying

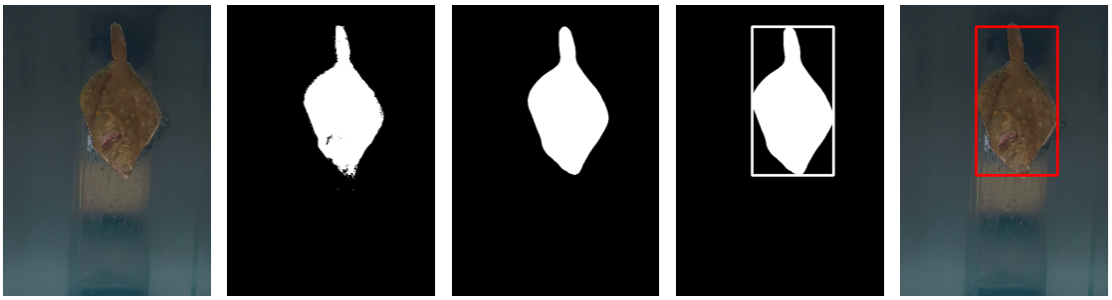
De tweede methode werkt op het principe van chroma-keying. Deze techniek wordt gebruikt bij bijvoorbeeld een green screen. Hierbij worden bepaalde kleuren gedetecteerd. Deze evaluatie wordt uitgevoerd per pixel. Als het kleur gedetecteerd wordt, wordt er een witte pixel opgeslagen. Als het kleur niet gedetecteerd wordt, wordt er een zwarte pixel opgeslagen. Deze informatie wordt opgeslagen in een aparte afbeelding, zoals te zien op de tweede foto op Figuur 3.3. Deze methode zal goed werken als er een hoog kleurcontrast is tussen de voor- en de achtergrond. In de video's zijn er bruine vissen te zien op een blauwe achtergrond. Dit is een hoog genoeg contrast om deze methode toe te passen. Ook zal er niet een enkele kleur gebruikt worden, maar een range van kleuren. Het resultaat van deze stap is dus een afbeelding in zwart en wit.

Blur en threshold

Als we de zwart-wit foto bekijken, zien we dat er enkele gaten en kleine stipjes te zien zijn. Een meer gunstige afbeelding zou een duidelijk gedefinieerde vorm zijn. Om dit te benaderen zullen we de afbeelding blurren en vervolgens thresholden. Blurren of vervagen betekent dat we kleine gaten en kleine stipjes zullen verspreiden. Vervolgens zullen we thresholden. Dit houdt in dat elke pixel boven een bepaalde waarde wit zal worden. Andersom zal de pixel zwart worden. Deze extra stappen hebben het effect dat gaten worden opgevuld en kleine stipjes worden weggewerkt. We bekomen een duidelijk gedefinieerde vorm zoals te zien op de derde foto op Figuur 3.3.

Bounding box

OpenCV, een open source beeldverwerking library, bezit een functie die een bounding box kan tekenen rond een zwart-wit afbeelding. Dit werkt het best op foto's die duidelijk gedefinieerde vormen bezit. Onze voorbereidde afbeeldingen zijn dus makkelijk te verwerken door deze functie. We verkrijgen een rechthoek waar de software een vis detecteert. Dit proces wordt afgebeeld op de laatste 2 foto's op Figuur 3.3.



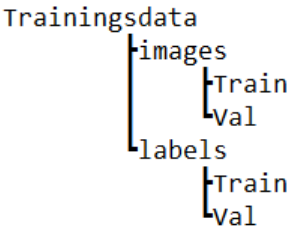
Figuur 3.3: Dataset creatie

Rand

De bounding box is soms net wat te klein, waardoor de neus of de staart van de vis net niet afgebeeld is. Dit kan vermeden worden door een kleine rand toe te voegen aan de uiteindelijke bounding box.

Naverwerking

Het bekomen resultaat moet vervolgens worden naverwerkt zodat deze kan gebruikt worden in de training. Voor YOLO moet de volledige frame opgeslagen worden met een tekst file waarin de class (soort), middelpunt van de bounding box en de afmetingen van de bounding box vermeld zijn. Deze files worden georganiseerd zoals te zien op Figuur 3.4.

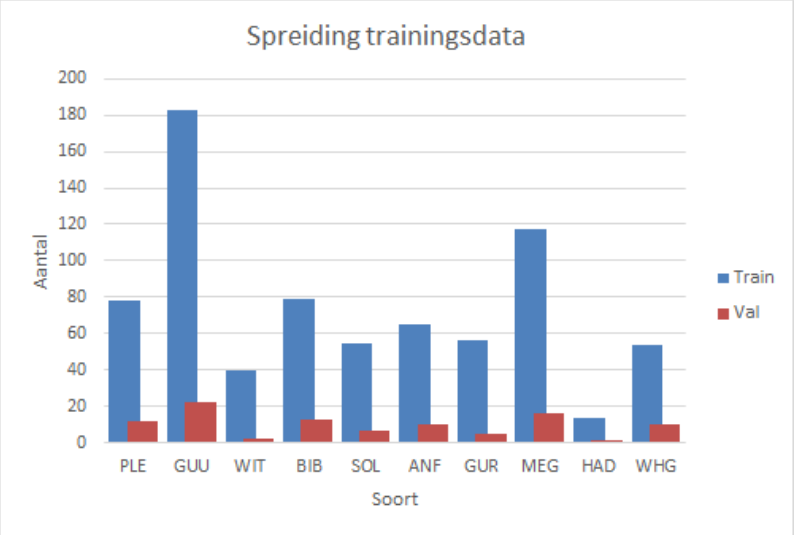


Figuur 3.4: Trainingsdata map structuur

3.2.3 Dataset Labelbox

De uitvoer van Labelbox is een .json file die vervolgens nog moet verwerkt worden om deze te gebruiken tijdens de training. Alles werd ingedeeld in het YOLO formaat om alles consistent te

houden. Door een miscommunicatie werden er ook afbeeldingen gelabeld met meerdere vissen per foto. Ook werden er afbeeldingen gelabeld waar net de neus te zien was. Deze afbeeldingen zijn niet ideaal om te trainen. Om de kwaliteit te garanderen van de trainingsdata, werd elke afbeelding apart gecontroleerd. Na de verwerking van alle data werd een spreiding bekomen, zoals te zien op Figuur 3.5. We zien dat er voor sommige soorten veel minder data beschikbaar is. Dit zal als resultaat hebben dat de modellen een bias zullen hebben naar de soorten waarbij er meer data beschikbaar is. In een ideaal geval zouden er hier per soort even veel afbeeldingen beschikbaar moeten zijn.



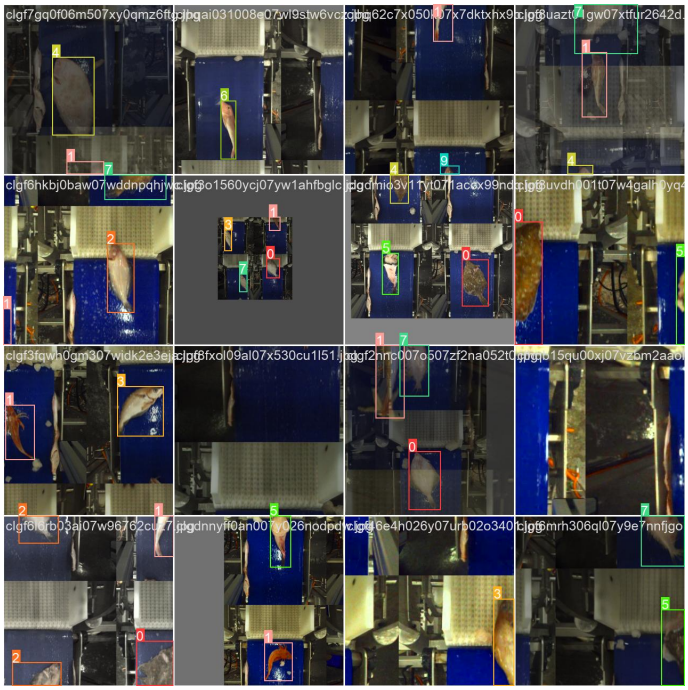
Figuur 3.5: Spreiding trainingsdata

3.2.4 Data augmentatie

Er kan gebruik gemaakt worden van data augmentatie om de dataset nog uit te breiden. Met data augmentatie wordt bedoelt dat de afbeeldingen en daarbij horende bounding boxes worden getransformeerd. Hierbij blijven de objecten op de afbeelding herkenbaar. Dit kan op veel manieren gebeuren. Er kan gespiegeld worden volgens verschillende assen, er kan geroteerd worden rond een bepaald punt en er kunnen verschillende afbeelding samengevoegd worden. Deze laatste methode werkt enkel bij toepassingen waar er meerdere bounding boxes kunnen verwerkt worden op 1 afbeelding. Op Figuur 3.6 is te zien hoe yolov5 deze augmentatie uitvoert.

3.2.5 Finale dataset

Uiteindelijk werd er gekozen om enkel de handgelabelde data via Labelbox te gebruiken. Er was al een grote onbalans aan trainingsdata tussen soorten, deze zou enkel sterker worden door de video data te verwerken. Ook leek het logischer om een model te maken die getraind en gebruikt wordt op afbeeldingen van 1 locatie. De uiteindelijke verdeling van trainingsdata is dus opnieuw te zien op Figuur 3.5. In totaal werden er 839 afbeeldingen in de trainingsdata verwerkt. Er werd een split van ongeveer 11 % validatie data en 89 % trainingsdata gehanteerd. Dit komt dus neer op 741 training afbeeldingen en 98 validatie afbeeldingen.



Figuur 3.6: Data augmentatie yolov5

3.3 Data learning

3.3.1 YOLO

Het eerste gekozen model zal de officiële implementatie van YOLO v5 zijn. Deze was zeer toegankelijk, aangezien de trainingsdata al voorbereid was in het YOLO formaat. Er zal getraind worden met een afbeeldingsgrootte van 300 pixels. Om te garanderen dat er lang genoeg getraind wordt, zal er voor 100 epochs getraind worden. Ook werd er gekozen voor het yolov5s netwerk. Dit is een iet wat kleinere versie van het yolov5 model. Een kleiner netwerk betekent een kortere trainingstijd. Een nadeel is echter dat er minder gedetailleerd kan getraind worden.

Yolov5 zal zelf de learn rate aanpassen naarmate dat het trainingsproces zich voortzet. Hier zal de learn rate dus telkens kleiner worden.

Yolov5 is gemakkelijk in gebruik en instelling. Het doel van deze thesis is echter wel een academische en in-depth benadering van machinevisie met AI. Er zitten zoveel features in het yolov5 pakket, dat het moeilijk wordt om elke stap van het programma te begrijpen en te tweaken. Daarom werd de beslissing gemaakt om een eigen implementatie te schrijven. Hierover wordt uitleg gegeven in het volgende onderdeel.

3.3.2 Resnet

Het tweede gekozen model zal resnet34 zijn. Deze keuze werd gemaakt omdat er veel literatuur te vinden was rond resnet. Ook was een voordeel dat er gebruik gemaakt kan worden van transfer learning. De trainingslus zal 2 verschillende modellen als uitvoer genereren. Het model met de laagste validatie loss zal opgeslagen worden als BEST... en het laatste model zal opgeslagen worden als LAST...

Om het effect te zien van de parameters worden er 8 verschillende modellen getraind. Deze zijn opgedeeld in 2 groepen. De eerste groep zal trainen op lagere resolutie afbeeldingen (LD) en de tweede groep zal trainen op hogere resolutie afbeeldingen (HD). In elke groep worden er 4 verschillende learn rates getest. Dit is te zien in Figuur 3.7.

Name	Epochs	Learn_rate	Batch_size	Image_size
LD1	100	0,0005	64	300
LD2	100	0,001	64	300
LD3	100	0,005	64	300
LD4	100	0,01	64	300
HD1	100	0,0005	16	500
HD2	100	0,001	16	500
HD3	100	0,005	16	500
HD4	100	0,01	16	500

Figuur 3.7: Verschillende training combinaties

3.4 Gebruikte software

3.4.1 Pytorch

Pytorch is een open source machine learning library. Dit is een hulpmiddel die de implementatie en validatie van neurale netwerken faciliteert. Deze library is gebaseerd op Torch. Torch is een machine learning framework die in LuaJIT geschreven is. Als dit compatibel is met de hardware, kan er gebruik gemaakt worden van de GPU om de matrix vermenigvuldigingen uit te voeren die bij een CNN gebruikt worden. Dit maakt de berekeningen een heel stuk sneller. Dit systeem wordt CUDA of Compute Unified Device Architecture genoemd. Ook zijn er al veel optimalisatie algoritmes geïmplementeerd in pytorch zelf. Een voorbeeld van deze optimalisaties is stochastic gradient descent. Deze optimalisaties zorgen ervoor dat het netwerk sneller zal leren. Er kan zeer snel een neuraal netwerk opgesteld worden met de NN module. Verder kan er nog geoptimaliseerd worden met DeepSpeed. [17]

In het volgende onderdeel zal de workflow van Pytorch en de NN module verduidelijkt worden. Er zal beschreven worden hoe er een neuraal netwerk kan worden opgebouwd, getraind en gevalideerd. Er werd gebruik gemaakt van een tutorial op towardsdatascience.com. [18]

Normaliseren

Eerst en vooral wordt er een functie gemaakt die de afbeeldingen zal normaliseren. Verschillende modellen hebben een andere conventie om dit uit te voeren. Deze functie zal dus afhankelijk zijn van het gewenste model. Hierbij zal er moeten vermeld worden wat de gewenste standaardafwijking en gemiddelde zijn.

Data Loaders

Vervolgens zal de data ingeladen worden. Dit gebeurt aan de hand van een data loader. Hier zal de data binnengeladen worden in batches. Er zal ook moeten worden vermeld hoe groot deze

batches zijn. Daarnaast wordt er geshuffeld om de dataset willekeurig te maken. In deze stap wordt ook de normalisatie functie gebruikt.

Model opstellen

In deze stap zal de NN module gebruikt worden om het netwerk op te stellen. Dit kan door elke laag zelf te definiëren. Een andere optie is om een voorgetraind model te gebruiken, hierbij kan er gebruik gemaakt worden van transfer learning. Ook zal er een loss functie worden opgesteld, deze functie bepaalt hoe correct het resultaat is van het netwerk. Dit zal gebruikt worden om te trainen.

Training

De trainingsloop zelf verloopt als volgt: eerst wordt de genormaliseerde afbeelding omgezet in een lijst van waarden. Vervolgens wordt de trainingsdata door het netwerk verwerkt en zal de loss berekend worden. Aan de hand van deze loss zal het netwerk leren door middel van backpropagation. Dit proces herhaalt zich voor de gewenste traningsdata. Er kan ook gekozen worden om een optimalisatie toe te passen. Dit zorgt ervoor dat het netwerk sneller zal trainen.

Validatie

Nadat de training is voltooid, zal er een validatie van het netwerk uitgevoerd worden. Hiervoor zal de validatiedata gebruikt worden. Er wordt telkens één voorbeeld ingelezen, deze wordt verwerkt door het netwerk. De voorspelling wordt dan vergeleken met het label van deze data. Als deze correct is, zal dit opgeteld worden bij de correct voorspelde data. Hieruit kan een percentage berekend worden die de nauwkeurigheid voorstelt.

3.5 Gebruikte hardware

De modellen zullen getraind worden op een Erazer Gaming Notebook Guardian X10. De CPU is een Intel Core i7 10750H @ 2.60GHz. Deze laptop beschikt over een krachtige NVIDIA GeForce RTX 2070 Super. Er wordt 8 van de 16 GB RAM gebruikt door het CUDA systeem.

3.6 Lengte meting

In dit deel worden de gebruikte agoritmes om de lengte meting uit te voeren verduidelijkt. Deze stap vindt plaats na de detectie en classificatie. De soort en bounding box worden hier dus gezien als gegeven. De bounding box maakt het gemakkelijker om de lengtebepaling correct uit te voeren.

Hierbij wordt er eerst een mask gemaakt om de vis te onderscheiden van de achtergrond. Dit kan gebeuren aan de hand van chroma keying. Hierbij is echter het probleem dat de waarden voor de keying handmatig ingesteld moeten worden om een juiste mask te krijgen. Er werden enkele pogingen gedaan om dit te automatiseren met klassieke machinevisie, maar dit zonder succes. Deze mask kan ook datagedreven achterhaald worden. Hierbij is er veel data nodig die gesegmenteerd is. Dit betekent dat er een veelhoek wordt getekend rond de voorgrond. Deze labels worden vervolgens gebruikt om een model te trainen die deze segmentering automatisch uitvoert. Op basis van deze segmentering kan een mask gemaakt worden.

Er worden enkele assumpties aangenomen over de vorm van de vis. De eerste assumptie is dat de vis langer is dan deze breed is. Ook wordt er aangenomen dat de vis niet zo gebogen ligt dat er per horizontale lijn, 2 punten van de ruggengraat worden aangetroffen.

3.6.1 SAM

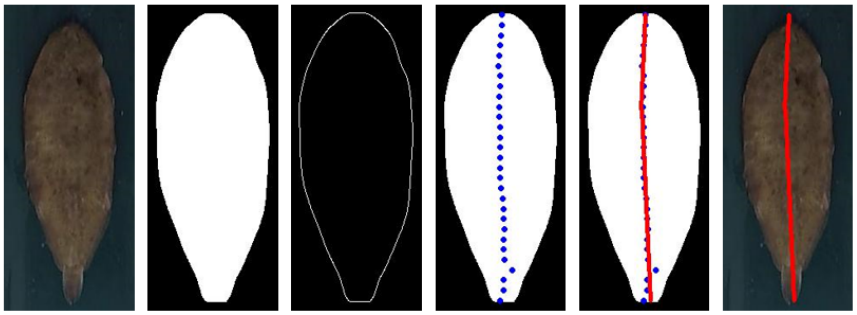
Om een mask te achterhalen kan er gebruik gemaakt worden van SAM. SAM of Segment Anything Model is een AI model, ontwikkeld door meta, om zaken te segmenteren in een afbeelding. Het is genoeg om een punt (coördinaten) op de afbeelding mee te geven. Hieruit kan er een mask afgeleid worden. Er kan ook gewerkt worden met een bounding box. Hieruit kan opnieuw een mask afgeleid worden. Het unieke aan SAM is dat deze geleerd heeft wat een algemeen ”object” is. Hierdoor is het dus mogelijk om objecten te segmenteren die het model nog nooit gezien heeft. Deze aanpak noemt zero-shot generalization. [19] Concreet kan SAM toegepast worden op de uitvoer van de detectie. Hier kan het punt in het midden van de bounding box genomen worden om zo de segmentatie hierop toe te passen. Een voorbeeld hiervan is te zien op Figuur 3.8. Deze afbeelding is gemaakt op de demo website rond SAM. Er kan ook gewerkt worden met een bounding box zelf, maar dit gaf slechte resultaten tijdens het testen. Deze methode zal dus verder niet gebruikt worden. Er bestaan libraries waar SAM geïmplementeerd is voor gebruik in python, dit maakt het makkelijk om deze te gebruiken in een eigen script.

Figuur 3.8: Segmentatie voorbeeld met SAM en 1 punt

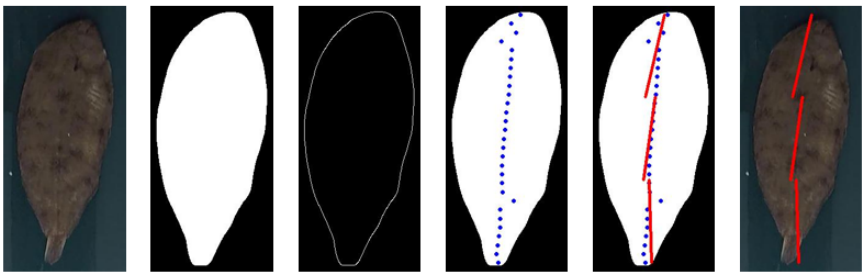
3.6.2 Gemiddelde methode

De eerste geteste methode werkt op basis van gemiddelden. Zoals te zien op Figuur 3.9, wordt er canny edge detectie toegepast op de mask. Hierbij is enkel de contour van de vorm aangeduid met witte pixels. Per horizontale lijn wordt er een gemiddelde genomen van waar zich de witte pixels bevinden. Bijvoorbeeld: als er witte pixels op x=10 en op x=90 te zien zijn, wordt de gemiddelde locatie berekend. Hier zou dit (90 + 10) / 2 = 50 zijn. Dit algoritme wordt voor een aantal lijnen op gelijke afstanden herhaalt. Dit levert een aantal blauwe punten op, te zien op Figuur 3.9. Deze blauwe punten worden opgedeeld in 3 subsets. Voor elk van deze subsets wordt

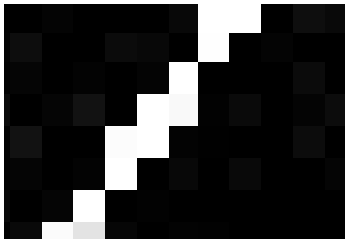
er lineaire regressie toegepast om de best passende rechte te vinden. Ten slotte worden deze lijnstukken getekend op de originele afbeelding. Voor symmetrische vissen, zoals Figuur 3.9, werkt dit algoritme vrij goed. Voor minder symmetrische vissen heeft dit algoritme nog wat gebreken. Een probleem die optrad was dat dit gemiddelde soms verkeerd lag, zoals op Figuur 3.10. Omdat de canny edge detection soms 2 pixels overlaat in de x-richting, zag het algoritme dit als 2 aparte punten die verrekend werden in het gemiddelde. Dit is te zien op Figuur 3.11. Hierdoor werden de lijnen niet juist berekend en is het resultaat niet zoals gewenst. Dit kon vermeden worden door een extra stap toe te voegen om deze gevallen weg te werken. Uiteindelijk werd er gekozen om een andere methode te proberen.



Figuur 3.9: Eerste voorbeeld gemiddelde methode



Figuur 3.10: Tweede voorbeeld gemiddelde methode

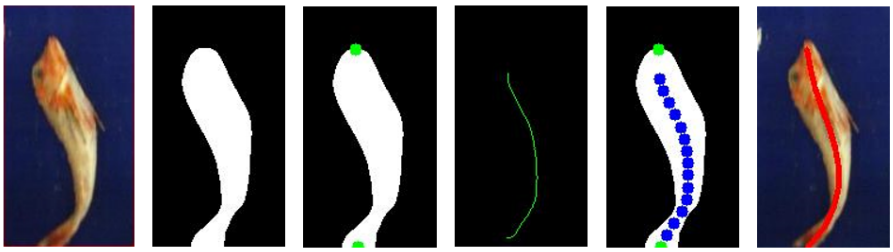


Figuur 3.11: Close up van de contour met 2 pixels horizontaal naast elkaar

3.6.3 Skelet methode

De tweede methode werkt op het principe van skeletonization. Dit is een techniek die toegepast wordt in klassieke machine visie om een dunne skelet structuur af te leiden van een binaire foto.

Hierbij wordt er recursief bepaald welke pixels er aan de rand van de vorm zitten. Deze worden vervolgens verwijderd op voorwaarde dat ze de keten niet breken. [20] Het volledige proces is te zien op Figuur 3.12. Eerst wordt de mask achterhaald door middel van SAM. Deze mask wordt ook geblurd en gethreshold om dezelfde redenen als bij de verwerking van de video data. Vervolgens wordt de bovenste en onderste rij gezocht, waar er witte pixels te vinden zijn. Ook wordt de gemiddelde x positie berekend op deze rij. Dit zorg ervoor dat dit punt in het midden van de vis ligt. Deze worden aangeduid met 2 groene punten. Vervolgens wordt de skeletonization toegepast op de mask. Dit geeft een lijn als resultaat die de vorm van de vis volgt. Deze lijn wordt gesampled op verschillende y posities. Dit wordt afgebeeld als de blauwe punten. Hierna kunnen we alle gevonden punten gebruiken om een veelterm regressie toe te passen. Het resultaat van dit algoritme is een kromme die de ruggengraat van de vis volgt. Praktisch wordt deze opgedeeld in 10 lineaire stukken waarvan de lengte makkelijk kan berekend worden. Deze 10 lengtes worden opgeteld om de totale lengte van de vis in pixels weer te geven.



Figuur 3.12: Voorbeeld skelet methode

3.6.4 Pixel coördinaten naar carthesische coördinaten

Het resultaat van de lengte bepaling tot nu toe zijn een aantal punten die de ruggengraat van de vis voorstellen. Deze moeten eerst nog individueel omgezet worden naar carthesische coördinaten. Om dit te doen zullen we de totale camera matrix nodig hebben. Dit is een matrix die omschrijft hoe carthesische coördinaten afgebeeld worden op de camera sensor en zo dus de pixel coördinaten. De totale camera matrix kan opgedeeld worden in 2 delen, de intrinsieke en de extrinsieke camera matrix. De extrinsieke camera matrix stelt de transformatie voor om van het wereld frame naar het camera frame om te vormen. Deze kan opgesteld worden door de positie en de rotatie van de camera ten opzichte van het wereld frame op te meten. De intrinsieke camera matrix stelt de transformatie voor om van 3D camera frame naar 2D pixel frame om te vormen. Om deze matrix op te stellen zal de focale lengte en de pixel dichtheid van de camera achterhaald moeten worden. Dit is in de data sheet van de camera vermeld.

Om deze matrices te praktisch te achterhalen, zal er een kalibratie uitgevoerd worden. Dit gebeurt aan de hand van een kalibratie tool. Deze kalibratie tool is een schaak patroon met gekende afmetingen. De hoekpunten worden gedetecteerd met klassieke machine visie. Deze hoekpunten worden vervolgens gebruikt om een stelsel vergelijkingen op te stellen. Deze vergelijkingen worden gebruikt om de totale camera matrix te achterhalen.

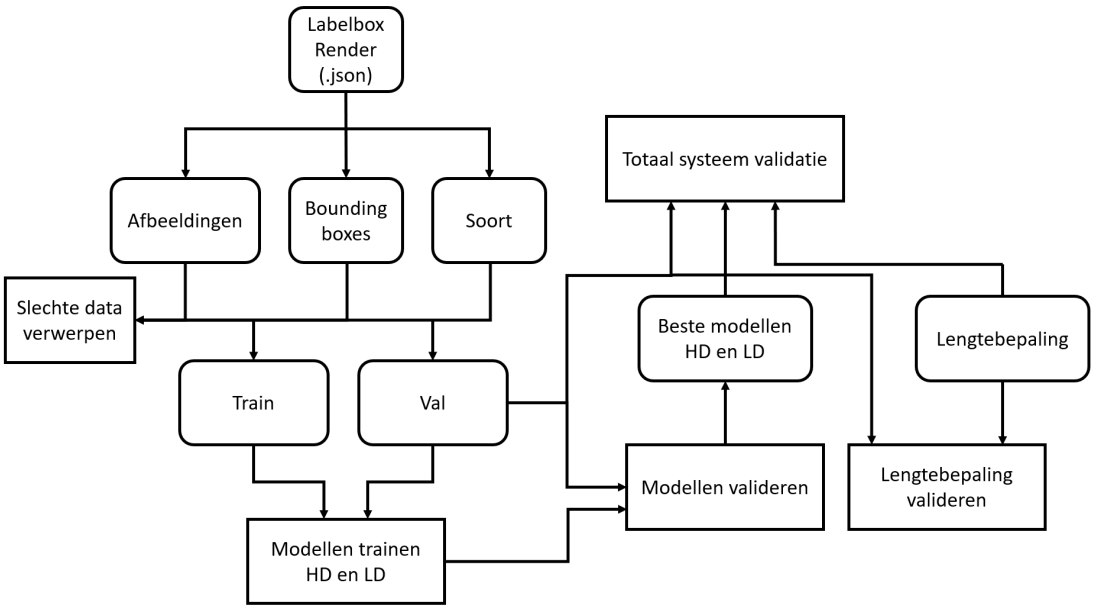
In een simpelere opstelling, zoals in deze case, kan er ook gewerkt worden met de focale lengte en de pixel grootte. [21]

3.6.5 Validatie

Omdat er geen lengtedata beschikbaar is voor de trainingsdata zal het algoritme visueel gevalideerd worden. Er zullen afbeeldingen uit de validatie dataset gebruikt worden als invoer. Hierbij wordt elk resultaat opgedeeld in 4 categorieën. De eerste categorie zullen de afbeeldingen zijn waar de ruggengraat juist gedetecteerd is. Afbeeldingen waar de ruggengraat juist gedetecteerd, is maar net iets te kort of te lang is, zullen in de tweede categorie geplaatst worden. In de derde categorie zullen de afbeeldingen waar de ruggengraat volledig verkeerd gedetecteerd is geplaatst worden. Ook zal er een vierde categorie zijn waar de afbeeldingen die resulteren in een error geplaatst worden.

3.7 Flowchart

In Figuur 3.13 wordt een flowchart afgebeeld van de volledige dataverwerking. Hier wordt ook verduidelijkt welke validaties er zullen toegepast worden. De modellen en de lengte meting zullen elk apart worden gevalideerd. Vervolgens zal ook het totale systeem gevalideerd worden. Hierbij wordt het beste HD en LD model gekozen als het gebruikte netwerk.



Figuur 3.13: Flowchart data

4 Resultaten

4.1 Detectie en classificatie

Eerst zullen de resultaten van het YOLO netwerk besproken worden. Vervolgens zullen de resultaten van de resnet netwerken besproken en vergeleken worden.

4.1.1 Yolo

Trainingstijd

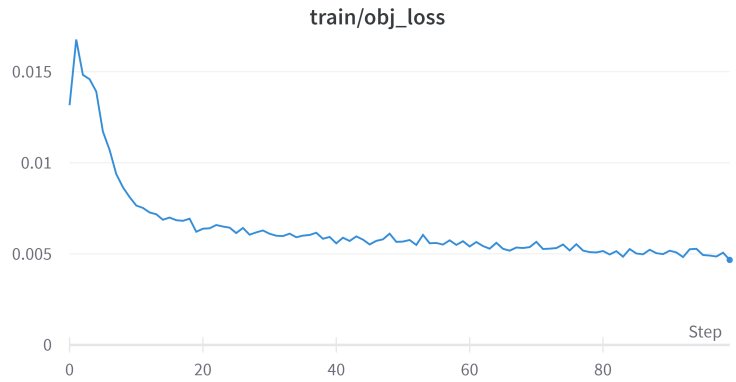
De totale trainingstijd van het yolov5 netwerk bedraagt 3,1 uur of 186 minuten. De training maakte geen gebruik van het CUDA systeem, waardoor deze enkel gebruik maakte van de CPU. Het is dus mogelijk om de training substantieel sneller te maken door dit op te lossen. De verdere resultaten zouden exact hetzelfde moeten zijn, aangezien het gebruik van CUDA enkel een invloed heeft op de trainingstijd.

Training loss

De grafiek van de training loss is te zien op Figuur 4.1. Deze grafiek zal eerst sterk dalen en vervolgens een plateau bereiken. Dit plateau wordt ongeveer bereikt aan epoch 30. Hierna zal er nog een kleine vooruitgang gemaakt worden. Dit is te wijden aan de learn rate die verlaagt. Naarmate de learn rate lager wordt, kunnen er fijnere details opgenomen worden in de training.

Precisie en recall

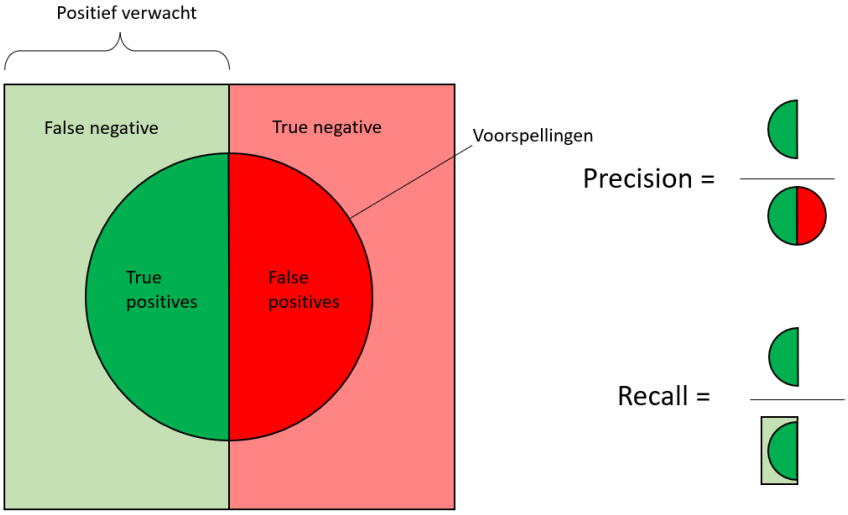
Met precisie wordt bedoeld hoeveel van de gedetecteerde objecten juist zijn. Dit zijn dus de true positives gedeeld door alle positives (positief resultaat voorspeld). Met de gebruikte parameters werd een precisie van 0,9689 behaald. Dit betekent dus dat 96,89% van de vissen juist gedetecteerd en geclassificeerd worden. Precisie kan dus vergeleken worden met juistheid.



Figuur 4.1: Training loss YOLO

Met recall wordt bedoeld hoeveel van de positieve tests worden gedetecteerd als positief? Dit zijn dus de true positives gedeeld door alle relevante data (positief resultaat verwacht). Er werd een maximale recall behaald van 0,8862. Recall kan vergeleken worden met compleetheid.

Deze concepten worden visueel weergegeven in Figuur 4.2.

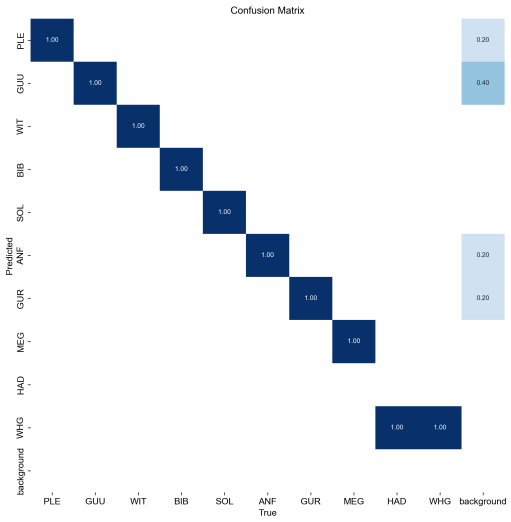


Figuur 4.2: Visualisatie precisie en recall

Confusion matrix

Een laatste methode om de performantie van een netwerk te meten is door middel van een confusion matrix. Hierbij wordt er een matrix opgesteld met op de ene as de werkelijke class en op de andere as de voorspelde class. Per afbeelding die gedetecteerd is, zal 1 opgeteld worden bij de cel die overeenkomt met de werkelijke class en de voorspelde class. We verwachten dat dit een diagonaal benadert. Deze is te zien op Figuur 4.3. We zien dat een groot deel juist wordt voorspeld. Echter is te zien dat de classificatie van de soort HAD incorrect verliep. Deze worden

telkens geclassificeerd als WHG. Er zijn zeer weinig afbeeldingen waarop een vis met soort HAD te zien is. Dit kan de reden zijn waarom er verkeerd wordt geclassificeerd bij de soort HAD. Het uiterlijk van WHG en HAD is ook vrij gelijkaardig. Ook wordt een achtergrond nooit juist geclassificeerd. Dit kan te maken hebben met het feit dat er geen pure achtergrond afbeeldingen in de dataset te vinden waren.



Figuur 4.3: Confusion matrix YOLO

Voorbeelden

Op Figuur 4.4 zijn enkele voorbeelden van de detectie en classificatie te zien. Deze resultaten zijn praktisch perfect. Er is geen enkele afbeelding te zien waar de vis niet correct gedetecteerd of geclassificeerd wordt. De staart en neus zijn allebei meegenomen in de bounding box.

4.1.2 Resnet

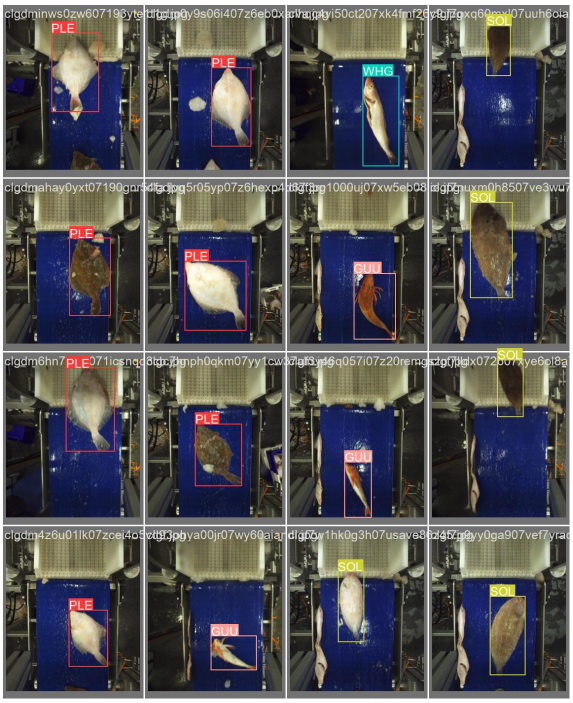
In dit onderdeel zullen alle HD en LD netwerken worden gevalideerd. Uiteindelijk zullen het beste HD en LD netwerk gekozen worden.

Trainingstijd

Op Figuur 4.5 kan een duidelijk verschil gezien worden in trainingstijden. De hogere resolutie groep duurt gemiddeld 185% langer om te trainen dan de groep met een lagere resolutie.

Training loss

Er wordt verwacht dat tijdens het trainen de training loss kleiner wordt. Op Figuur 4.6 is het verloop te zien van de LD groep tijdens het trainen. Er is te zien dat elk model sterk daalt in het begin en vervolgens afvlakt. Het ziet er naar uit dat LD1 de meest constante daling ondervindt. De eindwaarden van alle modellen zijn sterk gelijkaardig.



Figuur 4.4: Voorbeeld resultaat YOLO

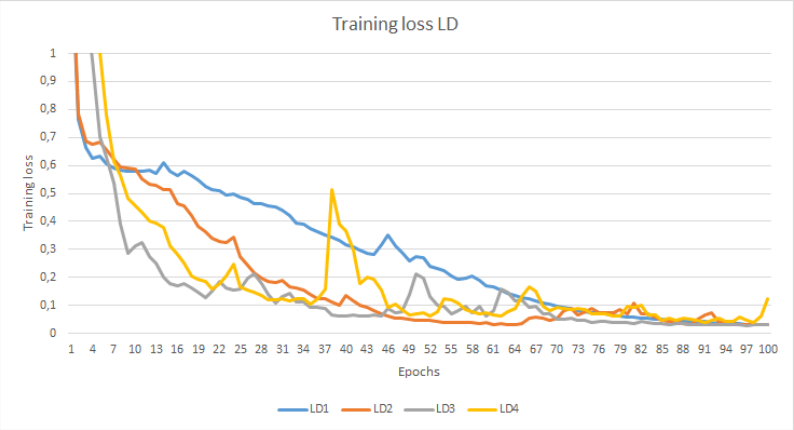
Name	time	
LD1	14,55	Minutes
LD2	14,48	Minutes
LD3	13,99	Minutes
LD4	14,05	Minutes
HD1	40,74	Minutes
HD2	40,76	Minutes
HD3	41,04	Minutes
HD4	40,04	Minutes

Figuur 4.5: Trainingstijden

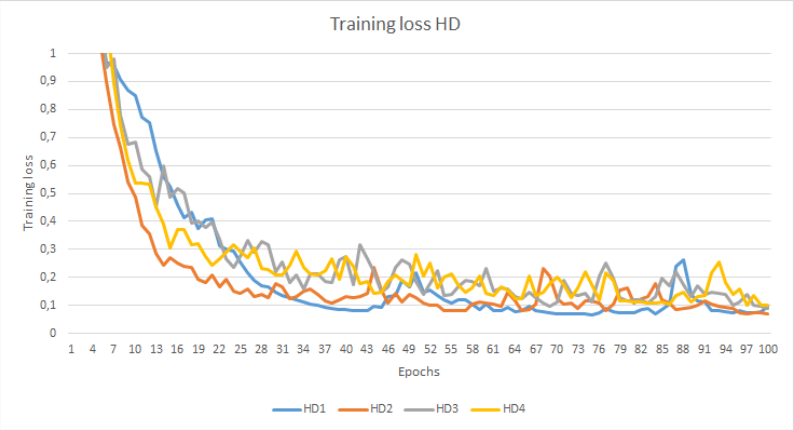
Op Figuur 4.7 is het verloop te zien van de HD groep. Deze is wat ruwer dan de LD groep. Dit komt deels omdat er meer informatie is om te verwerken. Door gebruik te maken van het BEST... netwerk kunnen deze pieken worden genegeerd. Er is ook te zien dat, in vergelijking met de LD groep, elk netwerk gelijkaardig traint. Er zijn geen grote verschillen in training loss voor de HD groep onderling. Opnieuw zijn de eindwaarden voor alle HD modellen gelijkaardig.

Validatie accuracy

Na het trainen werd er een steekproef uitgevoerd om een spreiding te krijgen van de verschillende validatie metrics. De eerste die besproken wordt is de validatie accuracy. In de resnet implementatie werd deze berekend als volgt. Het deel van de validatie dataset die een correcte voorspelling van



Figuur 4.6: Training loss bij trainen LD



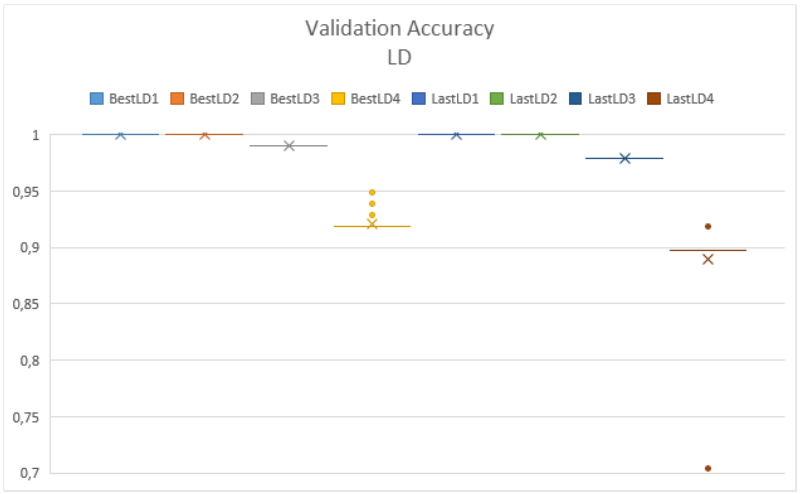
Figuur 4.7: Training loss bij trainen HD

de class krijgt is gelijk aan de validatie accuracy. Figuur 4.8 geeft de boxplots van deze steekproef weer. Er wordt vergeleken tussen de BEST en LAST van de 4 LD modellen. Binnen deze 2 groepen staan de modellen gesorteerd volgens oplopende learn rate. Zowel de BEST als LAST versies van LD1 en LD2 behaalden hier het beste resultaat, door op elke test een score van 100% te behalen. Algemeen is te zien dat de LAST modellen een slechtere performantie hebben dan de BEST modellen.

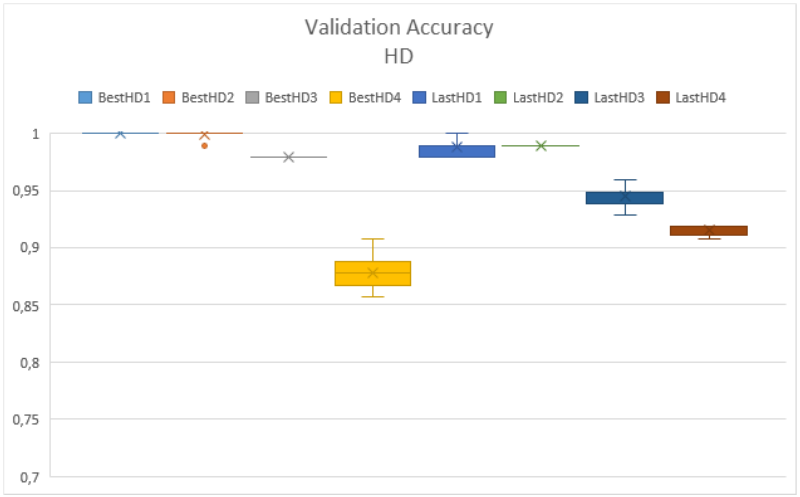
Op Figuur 4.9 is hetzelfde te zien maar dan van de HD groep. Hier gaf BestHD1 het beste resultaat. Opnieuw kan er geconcludeerd worden dat de BEST modellen een beter resultaat geven dan de LAST modellen.

Validatie loss

Omdat de validatie accuracy enkel een beeld geeft op hoe goed het model de class kan voorspellen geeft dit een vertekend beeld. De bedoeling is nog altijd om de class en de bounding box te voorspellen. Met de bounding box werd er dus geen rekening gehouden. Dit is wel het geval bij de validatie loss. Uit Figuur 4.10 concluderen we dat BestLD2 het beste resultaat geeft. Dit betekent dat voor LD modellen een lagere learn rate niet per se een beter resultaat vertoont. Er is dus een



Figuur 4.8: Validatie accuracy LD



Figuur 4.9: Validatie accuracy HD

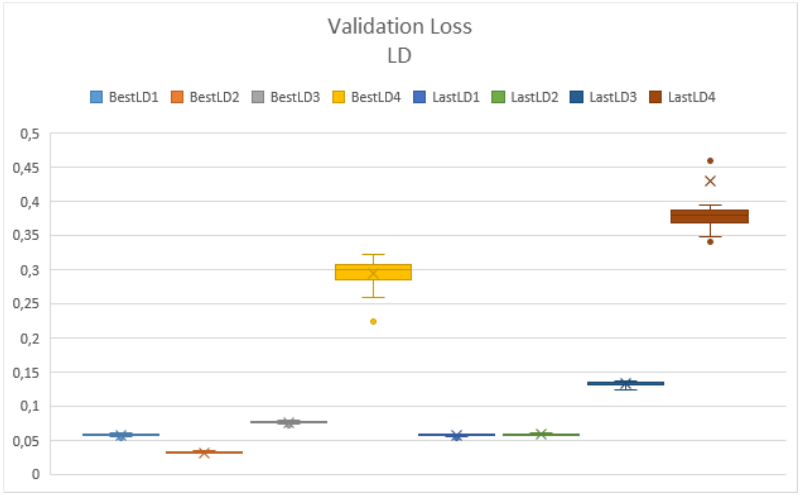
optimaal punt waar de resultaten het beste zijn. Opnieuw zijn de LAST modellen minder performant dan de BEST modellen.

Op Figuur 4.11 staan de validatie losses van de HD groep. Er kan worden geconcludeerd dat BestHD1 de beste resultaten geeft. Voor HD modellen geven lagere learn rates een beter resultaat. Opnieuw zijn de LAST modellen minder performant dan de BEST modellen.

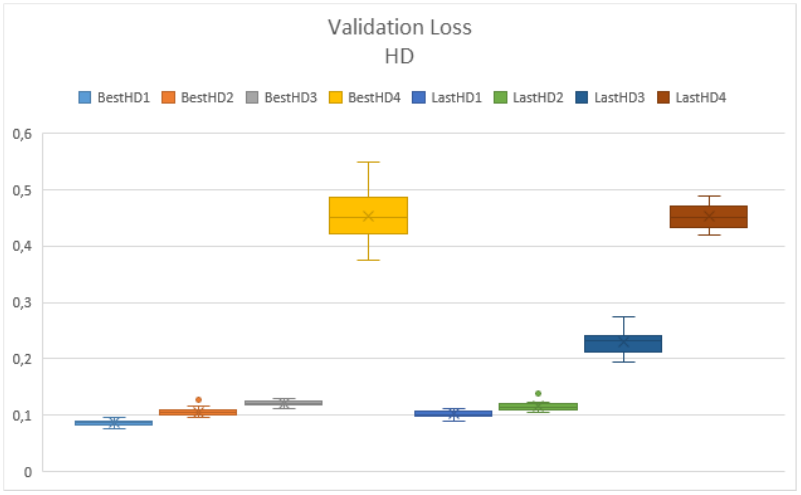
Confusion matrix

Net zoals bij het YOLO model kan hier een confusion matrix opgesteld worden. Op Figuur 4.12 is de confusion matrix te zien van het model BestLD2. Er is te zien dat de soort BIB de meeste problemen geeft. Verder zijn er ook elk 1 afbeelding van SOL en WHG die een verkeerde class zijn toegewezen. Het model kan dus vrij goed een afbeelding classificeren.

Op Figuur 4.13 is de confusion matrix te zien van het model BestHD1. Dit model heeft elke afbeelding in de juiste categorie geplaatst. Hierdoor krijgt BestHD1 dus een perfecte score op



Figuur 4.10: Validatie loss LD



Figuur 4.11: Validatie loss HD

		BestLD2									
		Werkelijke soort									
		PLE	GUU	WIT	BIB	SOL	ANF	GUR	MEG	HAD	WHG
Voorspelde soort	PLE	12	0	0	0	0	0	0	0	0	0
	GUU	0	22	0	1	1	0	0	0	0	0
	WIT	0	0	2	0	0	0	0	0	0	0
	BIB	0	0	0	6	0	0	0	0	0	0
	SOL	0	0	0	0	6	0	0	0	0	0
	ANF	0	0	0	0	0	10	0	0	0	0
	GUR	0	0	0	0	0	0	5	0	0	0
	MEG	0	0	0	0	0	0	0	16	0	0
	HAD	0	0	0	5	0	0	0	0	1	1
	WHG	0	0	0	1	0	0	0	0	0	9

Figuur 4.12: Confusion matrix BestLD2

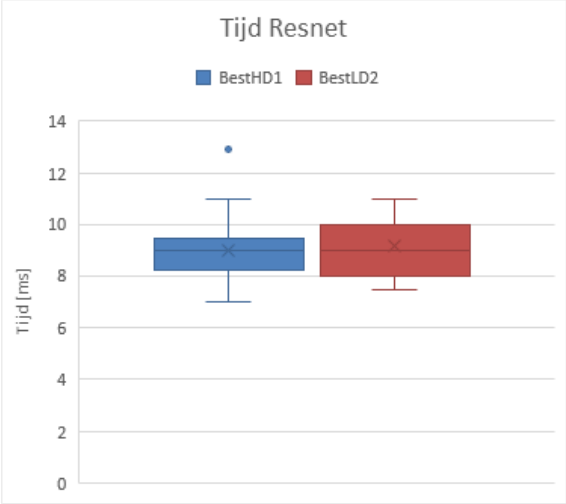
deze test.

		BestHD1									
		Werkelijke soort									
		PLE	GUU	WIT	BIB	SOL	ANF	GUR	MEG	HAD	WHG
Voorspelde soort	PLE	12	0	0	0	0	0	0	0	0	0
	GUU	0	22	0	0	0	0	0	0	0	0
	WIT	0	0	2	0	0	0	0	0	0	0
	BIB	0	0	0	13	0	0	0	0	0	0
	SOL	0	0	0	0	7	0	0	0	0	0
	ANF	0	0	0	0	0	10	0	0	0	0
	GUR	0	0	0	0	0	0	5	0	0	0
	MEG	0	0	0	0	0	0	0	16	0	0
	HAD	0	0	0	0	0	0	0	0	1	0
	WHG	0	0	0	0	0	0	0	0	0	10

Figuur 4.13: Confusion matrix BestHD1

Proces tijd

Op Figuur 4.14 zijn boxplots te zien die de proces tijd weergeven voor BestHD1 en BestLD2. Hieruit kan geconcludeerd worden dat er geen duidelijk verschil in detectie tijd is. Een LD netwerk is dus niet sneller dan een HD netwerk. Dit is te verwachten aangezien enkel de input laag kleiner is bij een LD netwerk. De berekeningen gebeuren in de tussenlagen die dus hetzelfde zijn bij beide netwerken.



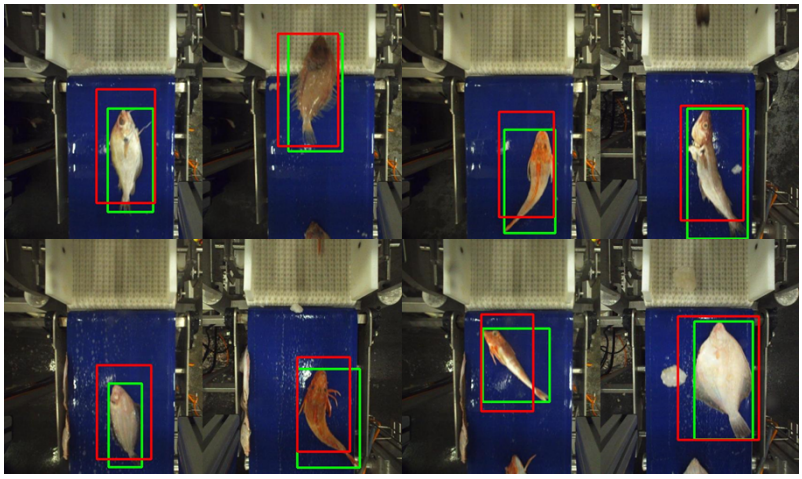
Figuur 4.14: Proces tijd BestHD1 en BestLD2

Voorbeelden

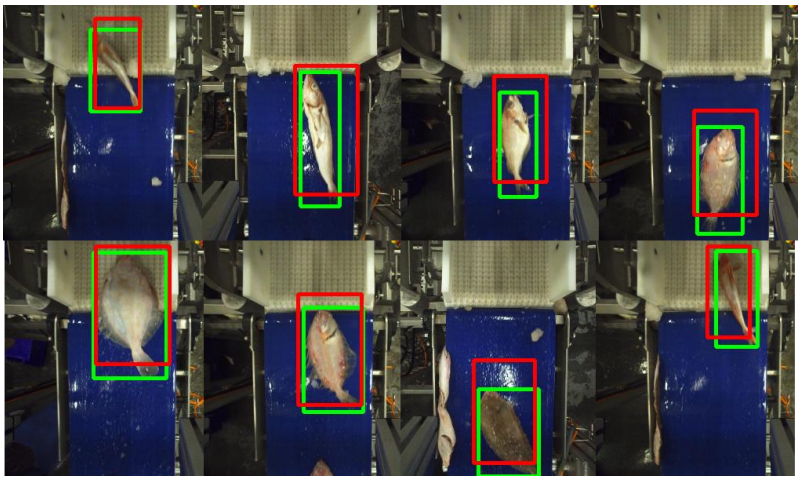
Op Figuur 4.15 zijn enkele voorbeelden te zien van de detectie van het model BestHD1. Deze afbeeldingen komen uit de validatie dataset. Hierop is de groene rechthoek de manueel getrokken bounding box. De rode rechthoek is de detectie die het model heeft bekomen. Er is te zien dat de staart een moeilijk punt is. Deze wordt soms niet meegenomen in de bounding box. De neus wordt meestal wel goed meegenomen.

Op Figuur 4.16 zijn enkele voorbeelden te zien van de detectie van het model BestLD2. Hierbij kunnen gelijkaardige conclusies getrokken worden zoals bij het vorige deel. De staart detecteren blijft het moeilijkste deel van de detectie. Opnieuw wordt de neus meestal wel meegenomen in de

bounding box.



Figuur 4.15: Voorbeelden detectie BestHD1



Figuur 4.16: Voorbeelden detectie BestLD2

4.1.3 Vergelijking

Uit de confusion matrix kunnen we concluderen dat het netwerk BestHD1 het best scoort van zowel de resnet modellen als het yolo model. Dit gaat wel enkel over de classificatie en zegt niets over de locatie van de voorspelde bounding box. Omdat Yolo en resnet de loss op verschillende manieren berekenen, is een verdere vergelijking maken moeilijk. Aan de hand van de voorbeelden is te zien dat het YOLO model de staarten wel meeneemt in de bounding box. Hierdoor kan deze gezien worden als een kwalitatievere detectie dan de resnet modellen.

4.2 Lengte meting

In dit onderdeel zal het algoritme gevalideerd worden die de lengte opmeet. Hierbij zal de bounding box gebruikt worden zoals aangeduid door de trainingsdata. Er zullen enkel validatie afbeeldingen gebruikt worden om de resultaten later te kunnen vergelijken. De afbeeldingen worden opgedeeld in 4 categorieën, zoals beschreven bij de methodologie. Vervolgens zal de proces tijd ook berekend en besproken worden.

4.2.1 Visuele validatie

Op Figuur 4.17 zijn de resultaten te zien van deze validatie. Er wordt een succes rate behaald van 74%. De 11% waarbij de meting te kort of te lang is wordt vooral veroorzaakt door SAM die de staart niet meeneemt in de mask. Hierdoor stopt de lijn dus te vroeg. Op 9% van de afbeeldingen is een ruggengraat detectie te zien die een verkeerde vorm heeft. Dit is te wijden aan de vis die niet in het midden van de bounding box ligt. Hierdoor zal SAM de achtergrond segmenteren in plaats van de voorgrond. Dit geeft dus een verkeerd resultaat. De 5% die een error gaf kan opnieuw verklaard worden door een verkeerde mask. Een juiste mask bepalen blijkt dus de moeilijkste stap in het algoritme.

74%	Correct
11%	Start/end wrong
9%	Wrong shape
5%	Error

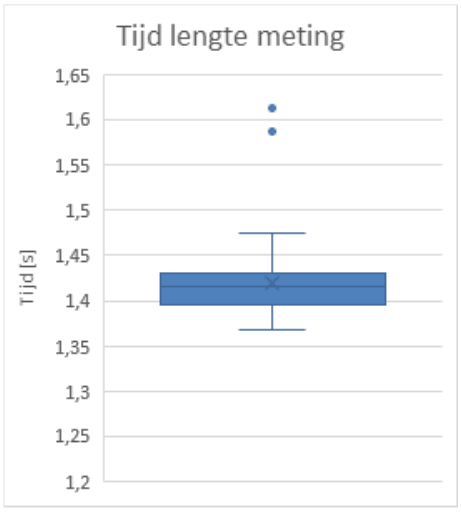
Figuur 4.17: Resultaten validatie lengtebepaling

4.2.2 Proces tijd

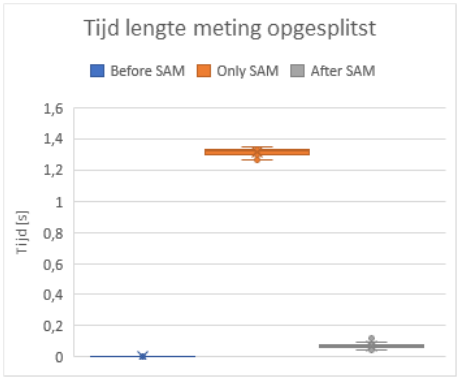
Op Figuur 4.18 is een boxplot te zien die de proces tijd weergeeft voor de lengtemeting. Er wordt een gemiddelde proces tijd van 1,42 seconden gehaald. Dit is vrij lang en zorgt ervoor dat het systeem mogelijk niet real time kan runnen. Het grootste deel van deze tijd is te wijden aan het genereren van de mask door SAM. Dit is te zien op Figuur 4.19 waar de proces tijd verder is opgesplitst. Er wordt een onderscheid gemaakt tussen de stappen voor SAM, SAM zelf en de stappen na SAM. In de stappen voor SAM wordt de afbeelding gecropt volgens de meegegeven bounding box. De stappen na SAM zijn degene die worden beschreven in de methodologie nadat de mask verkregen is.

4.3 Totaal systeem

In dit onderdeel zal het volledige systeem gevalideerd worden. Er wordt een onderscheid gemaakt tussen het systeem waar BestHD1 gebruikt wordt en waar BestLD2 gebruikt wordt. Hier worden alle validatie afbeeldingen door het volledige systeem verwerkt. Het resultaat wordt opnieuw visueel gevalideerd en opgedeeld in 4 categorieën.



Figuur 4.18: Proces tijd lengtemeting



Figuur 4.19: Proces tijd lengtemeting opgesplitst

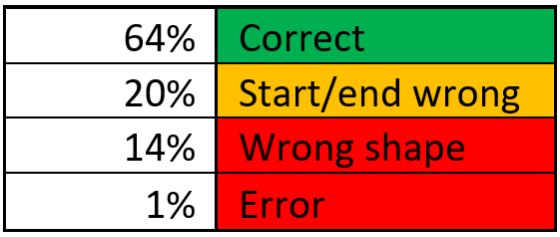
4.3.1 Visuele validatie

HD model

Op Figuur 4.20 zijn de resultaten te zien van de validatie. Hier werd gewerkt met het model BestHD1. Er wordt een succes rate van 64% behaald. We kunnen deze waarde vergelijken met de succes rate van de lengte meting zelf. Hierbij werd de optimale bounding box gebruikt. Dit betekent dat de behaalde 74% de best mogelijke score is met het gebruikte algoritme. Als we 64% delen door de maximale waarde van 74% bekomen we dat BestHD1 86% van de performantie uit het lengte meting algoritme haalt.

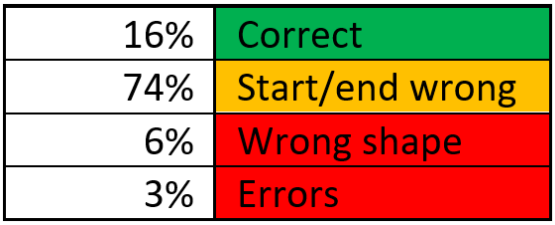
LD model

Op Figuur 4.21 zijn de resultaten van de validatie te zien. Hier werd gebruik gemaakt van het model BestLD2. Er werd een succes rate van 16% gehaald. Dit is vrij weinig. Tijdens het valideren was het duidelijk dat een groot deel van de bounding boxes de staart niet meenam. Dit resulteert in een gegarandeerde te korte meting. Hierdoor zijn 74% van de metingen te kort. Er werd slecht 22% van de performantie uit het lengte meting algoritme gehaald. Om hier toch betere resultaten uit te



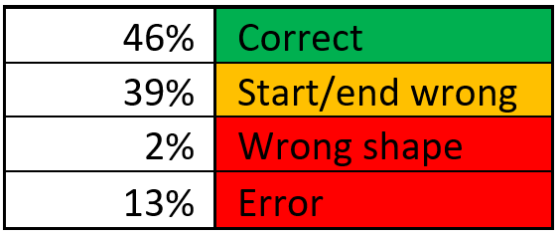
Figuur 4.20: Resultaten validatie totaal systeem met BestHD1

halen werd er besloten om een extra stap toe te voegen.



Figuur 4.21: Resultaten validatie totaal systeem met BestLD2

In deze extra stap werd een boord van 20 pixels breed toegevoegd aan de bounding box. Het belangrijke aan deze aanpak is dat het midden van de afbeelding gelijk blijft. Hierdoor ligt het punt die doorgegeven wordt aan SAM nog even veel op de vis. Nadat deze stap geïmplementeerd werd, werd de validatie opnieuw uitgevoerd. De resultaten zijn te zien op Figuur 4.22. Uiteindelijk werd een succes rate van 46% behaald. Dit is een verbetering van 288% in vergelijking met het LD model zonder de extra stap. Verder is er ook te zien dat het aantal errors gestegen is. Dit heeft als reden dat wanneer de afbeelding gecropt wordt, deze soms buiten de boorden van de originele afbeelding beland. Dit resulteert in een error. Dit probleem valt op te lossen, maar aangezien het HD model een hogere succes rate heeft zonder extra stappen lijkt de conclusie dat een LD model niet geschikt is voor deze toepassing.

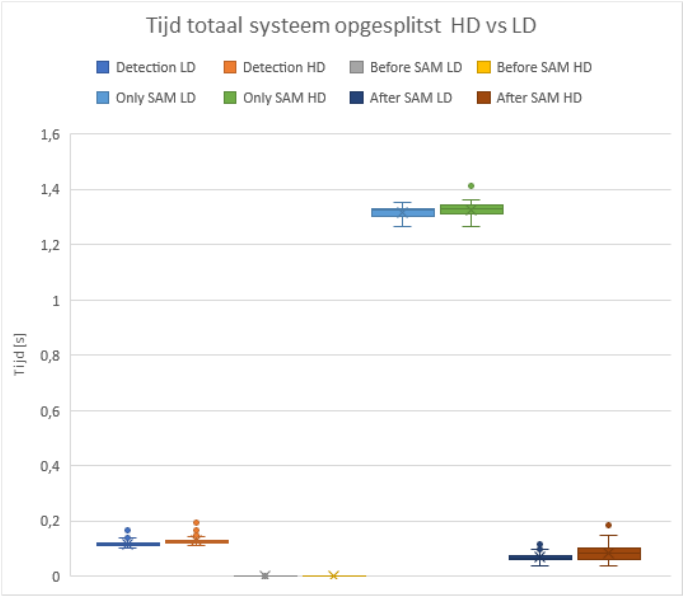


Figuur 4.22: Resultaten validatie totaal systeem met BestLD2 en extra stap

4.3.2 Proces tijd

Op Figuur 4.23 is de proces tijd per onderdeel te zien. Ook wordt er een vergelijking gemaakt tussen het gebruik van een HD of LD netwerk. Hier is te zien dat er een klein verschil is tussen deze twee mogelijkheden. De stappen van de lengte meting nemen dus heel marginaal toe in

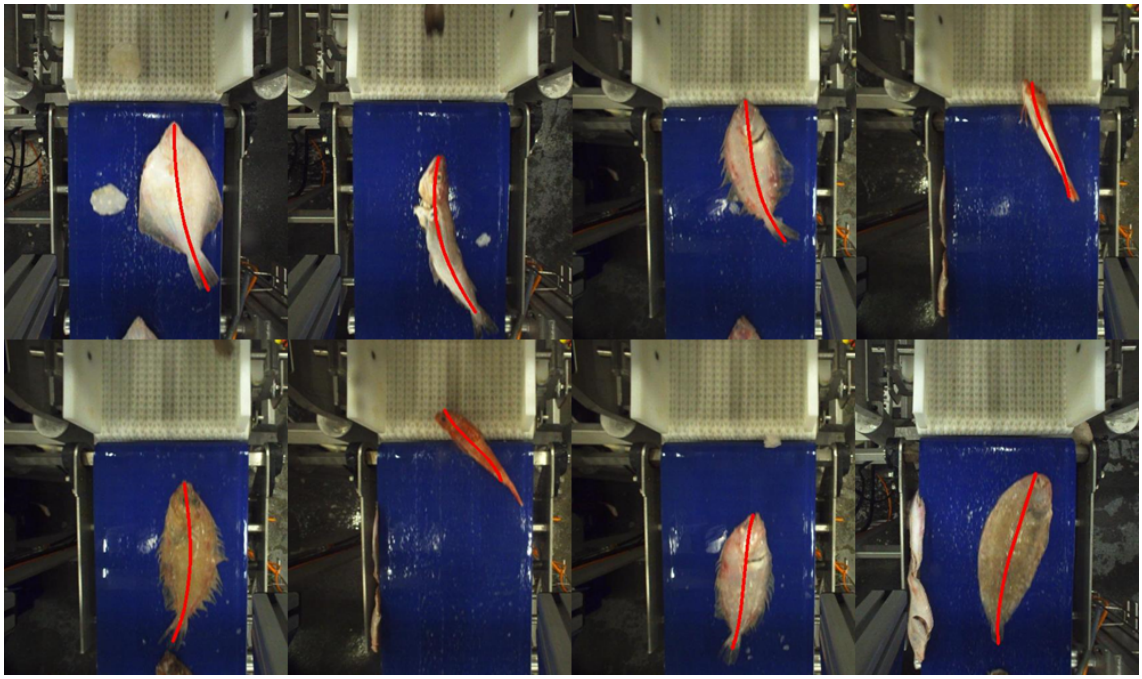
tijd als de afbeelding kleiner is. Dit effect is zeer miniem en kan dus moeilijk gezien worden als voordeel. SAM neemt in beide gevallen nog altijd het grootste deel aan tijd in beslag.



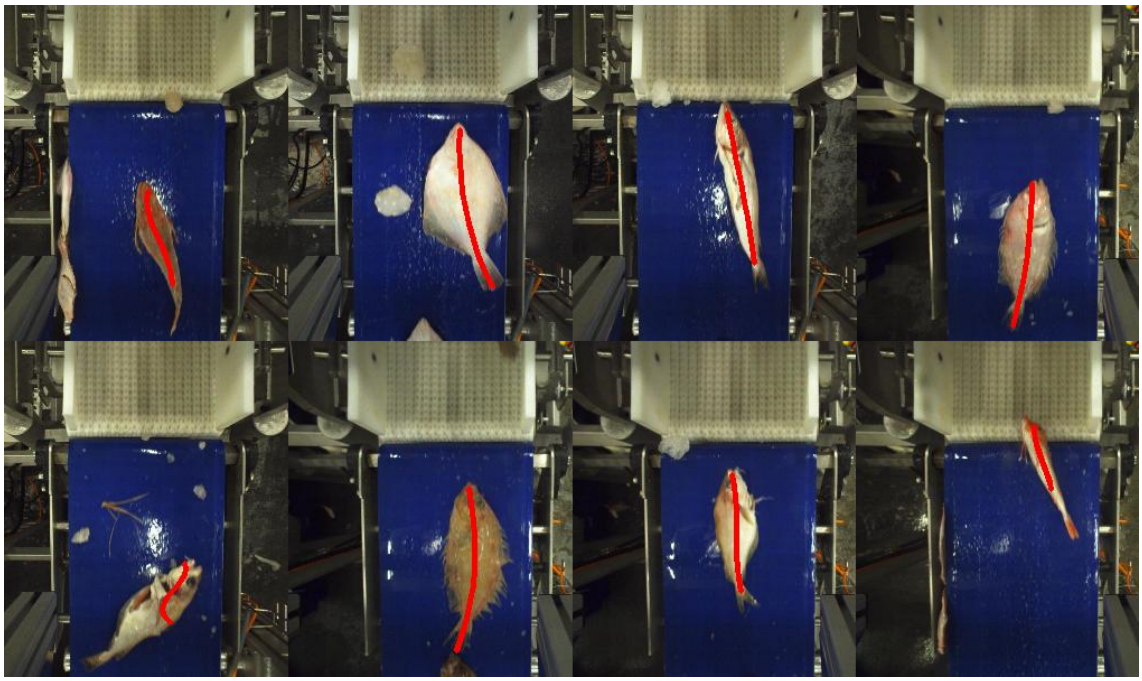
Figuur 4.23: Proces tijd totaal systeem opgesplitst HD vs LD

4.3.3 Voorbeelden

Op Figuur 4.24 worden enkele voorbeelden getoond van het totale systeem waarbij BestHD1 gekozen werd als model. Er is te zien dat op een groot aantal van de afbeeldingen de ruggengraat juist gedetecteerd wordt. Enkel wordt de staart in sommige gevallen niet meegenomen. Op Figuur 4.25 zijn opnieuw een reeks voorbeelden te zien. Hierbij werd het model BestLD2 gebruikt. Opnieuw wordt de staart hier soms niet meegenomen in de meting. Ook wordt in 1 voorbeeld de ruggengraat totaal verkeerd gedetecteerd. Dit is een voorbeeld van categorie 3 in de visuele validatie.



Figuur 4.24: Voorbeelden totaal systeem met BestHD1



Figuur 4.25: Voorbeelden totaal systeem met BestLD2

5 Conclusie

5.1 Trainingsdata

Een deel van de geleverde trainingsdata was niet conform met de conventies die het resnet nodig had om te trainen. Hierdoor moest een groot stuk van de data verworpen worden. Dit resulteerde in een kleinere en ongebalanseerde dataset. Optimaal zou dit een grote en gebalanseerde dataset zijn. Momenteel zal het netwerk een bias hebben naar bepaalde soorten waar meer trainingsdata beschikbaar is.

5.2 Detectie en classificatie

5.2.1 Resnet

Het model BestHD1 had een betere performantie dan het model BestLD2. Dit bleek nog duidelijker in de validatie van het totale systeem. Er is ook bijna geen voordeel aan een LD model in vergelijking met een HD model. Een LD model maakt een verwaarloosbaar snellere detectie dan een HD model. Enkel is een LD model sneller getraind. Aangezien dit een stap is die offline gebeurd heeft dit geen impact op de performantie van het systeem. Er wordt dus beter gewerkt met een HD model dan met een LD model. Uiteindelijk werd het doel om de vissen te detecteren en te classificeren behaald.

Het meenemen van de staart blijkt een moeilijk punt voor zowel LD als HD modellen. Dit heeft mogelijk als oorzaak dat er minder visueel contrast is tussen de half doorzichtige staart en de lopende band. Dit zou mogelijk opgelost kunnen worden door meer trainingsdata te voorzien.

5.2.2 Yolo

Yolo bracht zeer goede resultaten. Hierbij was de staart geen probleem. Er werd praktisch altijd een juiste detectie uitgevoerd. Enkel de classificatie stap verliep hier net wat minder. Hier was af

te leiden uit de confusion matrix dat yolo moeite had met de soort HAD. Dit had ook als reden dat er zeer weinig validatie data beschikbaar was voor deze soort. In het geval van HAD was er maar 1 validatie afbeelding.

5.3 Lengte meting

Het algoritme om de lengte te meten werkt goed as de mask correct is. Het genereren van deze mask is het deel waar de meeste fouten voorkomen. Als deze mask juister kan worden gegenereerd zal het systeem dus een betere performantie hebben. Uiteindelijk werd een maximale succes rate van 74% behaald.

Verder is SAM ook een traag algoritme. Deze stap neemt het meest tijd in van heel het systeem. Hierdoor is het systeem mogelijk te traag om dit real time te doen werken.

5.4 Totaal systeem

In het totale systeem was opnieuw duidelijk dat een HD model een hogere performantie had dan een LD model. Hier werd een maximale succes rate van 64% behaald.

Het totale systeem kan worden ingezet om de spreiding van de lengte per soort op te stellen. Hiermee is het hoofddoel bereikt.

5.5 Future work

De segmentatie is de stap waar het meeste fouten optraden. Dit zou mogelijk in de toekomst nog verbeterd kunnen worden. Ook zouden de resultaten positief beïnvloed worden moest er meer trainingsdata aanwezig zijn en moest deze beter verdeeld zijn.

Het resultaat van het totale systeem zou ook beter zijn als het yolo netwerk gebruikt wordt om de bounding box te voorspellen. In deze scriptie is dit niet gebeurd omdat de focus meer lag op de impact van de verschillende parameters van resnet.

6
Bibliography

[1] “Yolo object detection.” [Online]. Available: https://www.v7labs.com/blog/yolo-object-detection

[2] “Rcnn.” [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[3] “Nms en iou.” [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

[4] M. Islam, J. F. Ani, A. Rahman, and Z. Zaman, “Fake hilsa fish detection using machine vision,” *CoRR*, vol. abs/2201.02853, 2022. [Online]. Available: https://arxiv.org/abs/2201.02853

[5] “Different types of fish’s length measurement.” [Online]. Available: https://www.researchgate.net/figure/Different-types-of-fishs-length-measurement_fig1_335179001

[6] “Mer2-u3 series.” [Online]. Available: https://en.daheng-imaging.com/show-106-1985-1.html

[7] “Lcm-5mp-04mm-f2.0-1.8-nd1.” [Online]. Available: https://www.get-cameras.com/LENS-C-mount-5MP-4MM-F2.0-1/1.8-LCM-5MP-04MM-F2.0-1.8-ND1

[8] “Ilvo.” [Online]. Available: https://ilvo.vlaanderen.be/nl/organisatie-waarden-missie

[9] “Convolutie.” [Online]. Available: https://www.sciencedirect.com/topics/engineering/convolutional-layer#:~:text=A%20convolutional%20layer%20is%20the,and%20creates%20an%20activation%20map.

[10] “Retinanet.” [Online]. Available: https://paperswithcode.com/method/retinanet

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[12] “blacksuan19.” [Online]. Available: https://www.blacksuan19.dev/projects/fish-classification-with-pytorch-resnet/

[13] “How to measure a fish.” [Online]. Available: https://www.koaw.org/measuring-fishes

[14] D. White, C. Svellingen, and N. Strachan, “Automated measurement of species and length of fish by computer vision,” *Fisheries Research*, vol. 80, no. 2, pp. 203–210, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165783606001512

[15] D. L. Mingming Hao, Helong Yu, “The measurement of fish size by machine vision - a review,” 2015. [Online]. Available: https://inria.hal.science/hal-01614170/file/434298_1_En_2_Chapter.pdf

[16] “Mer2-302-56u3c.” [Online]. Available: https://www.get-cameras.com/USB3.0-Camera-3MP-Color-Sony-IMX265-MER2-302-56U3C

[17] “Pytorch.” [Online]. Available: https://github.com/pytorch/pytorch/releases/tag/v1.13.0

[18] “Tutorial pytorch.” [Online]. Available: https://towardsdatascience.com/handwritten-digit-mnist-pytorch-977b5338e627

[19] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, “Segment anything,” *arXiv:2304.02643*, 2023.

[20] “Skeletonize.” [Online]. Available: https://scikit-image.org/docs/stable/auto_examples/edges/plot_skeleton.html

[21] M. Dr. ing. De Ryck, “Machinevision lesson 5 image formation.”

FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
CAMPUS BRUGGE
Spoorwegstraat 12
8200 BRUGGE, België
tel. + 32 50 66 48 00
iiw.brugge@kuleuven.be
www.iiw.kuleuven.be