

# Objectdetectie in sonardata met behulp van semi- en self-supervised learning.

Een verkennend onderzoek naar het toepassen van moderne leertechnieken op onderwaterbeeldvorming.

---

**Yoran Gyselen.**

Scriptie voorgedragen tot het bekomen van de graad van  
Professionele bachelor in de toegepaste informatica

**Promotor:** Mevr. C. Teerlinck

**Co-promotor:** Mevr. S. Duyck

**Instelling:** Exail Robotics Belgium

**Academiejaar:** 2024–2025

**Tweede examenperiode**

**Departement IT en Digitale Innovatie .**

**HO  
GENT**



# Woord vooraf

Voor u ligt mijn bachelorproef over het gebruik van semi- en self-supervised learning-technieken voor objectdetectie op sonarbeelden. Dit onderzoek richt zich op de vraag of dergelijke technieken het labelproces kunnen versnellen zonder significant verlies in nauwkeurigheid. Het doel is een efficiëntere methode ontwikkelen voor het verwerken van sonardata.

Mijn interesse in dit onderwerp ontstond vanuit een combinatie van mijn passie voor machine learning en de uitdagingen die ik tijdens mijn stage tegenkwam. Het verwerken en labelen van sonardata bleek een tijdrovend en arbeidsintensief proces te zijn. Dit gaf me het idee om innovatieve methoden te verkennen om dit te optimaliseren. Semi- en self-supervised learning boden een veelbelovende oplossing, en ik was benieuwd of deze technieken in de praktijk daadwerkelijk een verschil konden maken.

Het schrijven van deze bachelorproef was een enorm leerzaam, maar uitdagend proces, waarin ik veel heb bijgeleerd over machine learning en de praktische toepassingen ervan binnen de industrie. Dit onderzoek zou niet mogelijk zijn geweest zonder de steun en begeleiding van verschillende mensen, aan wie ik graag mijn dank wil uitspreken.

Allereerst wil ik mijn promotor, mevr. Chantal Teerlinck, bedanken voor de goede begeleiding, feedback en inzichten tijdens dit traject. Haar ervaring en ondersteuning hebben me geholpen om dit onderzoek in de juiste richting te sturen. Daarnaast wil ik mijn co-promotor, mevr. Stefanie Duyck, bedanken voor haar betrokkenheid, kennis en praktische inzichten vanuit de bedrijfswereld, wat een belangrijke meerwaarde vormde voor dit onderzoek.

Ook wil ik mijn dank uitspreken aan Exail Robotics Belgium voor de kans om mijn bachelorproef binnen hun organisatie uit te voeren. De toegang tot zeer waardevolle resources en de begeleiding vanuit het team hebben een cruciale rol gespeeld in het realiseren van dit onderzoek. Tot slot wil ik mijn familie en vrienden bedanken voor hun steun en aanmoediging gedurende mijn studietraject.

Ik hoop dat deze bachelorproef een bijdrage kan leveren binnen het domein van objectdetectie op sonarbeelden en machine learning in het algemeen.

# Samenvatting

Sinds de opkomst van krachtige AI- en deep learning-modellen is data uitgegroeid tot een essentiële en vaak beperkende factor in het ontwikkelingsproces. Waar eenvoudige modellen vaak kunnen volstaan met beperkte en eenvoudige datasets, vereisen complexere modellen – zoals die voor objectdetectie – steeds grotere en rijkere hoeveelheden gelabelde data. Dit vormt een belangrijk probleem in domeinen zoals sonarbeeldvorming, waar dergelijke datasets niet beschikbaar zijn als kant-en-klare bronnen en handmatige annotatie buitengewoon tijdsintensief en kostbaar is. Dit onderzoek richt zich daarom op de centrale vraag: hoe kunnen semi-supervised en self-supervised leermethoden het labelproces bij objectdetectie in sonardata versnellen, zonder significant verlies aan nauwkeurigheid?

Om deze vraag te beantwoorden is een experimenteel kader opgezet waarin drie benaderingen zijn onderzocht: een volledig supervised baseline gebaseerd op Faster R-CNN, een semi-supervised model met FixMatch, en een self-supervised strategie waarbij een BYOL-model wordt gepretraïnd en vervolgens gebruikt als backbone binnen een Faster R-CNN-architectuur. De experimenten zijn uitgevoerd op een publieke sonardataset bestaande uit 7600 gelabelde sonarbeelden. Voor de supervised baseline is het model getraind op verschillende hoeveelheden gelabelde data: 1%, 5%, 10%, 50% en 100%. De bijbehorende mAP-scores tonen een sterke daling in nauwkeurigheid naarmate de hoeveelheid gelabelde data afneemt, met resultaten variërend van 0.7717 (100%) tot slechts 0.2799 bij gebruik van 1% van de data.

In het semi-supervised scenario is FixMatch toegepast met 5% en 10% gelabelde data, terwijl de resterende data werd gebruikt als ongelabelde input. Deze aanpak resulteerde in mAP-scores van respectievelijk 0.6649 en 0.6828, wat duidelijk betere prestaties zijn dan het supervised model op dezelfde labelniveaus. Voor het self-supervised model werd BYOL gepretraïnd op de volledige dataset zonder labels. De representaties die dit opleverde zijn vervolgens geïntegreerd in Faster R-CNN, waarbij opnieuw 5% en 10% van de data gelabeld werd gebruikt voor training. Deze benadering leverde de hoogste nauwkeurigheid binnen de lage-labelscenario's, met mAP-scores van respectievelijk 0.6452 en 0.7230.

---

De resultaten van dit onderzoek tonen aan dat zowel semi-supervised als self-supervised technieken effectief zijn in het verminderen van de afhankelijkheid van handmatig gelabelde data, terwijl de modelprestaties grotendeels behouden blijven. Met name self-supervised pretraining via BYOL blijkt zeer waardevol in situaties met beperkte gelabelde data. Deze bevindingen bieden praktische aanknopingspunten voor het ontwikkelen van efficiëntere workflows in sonarbeeldanalyse, en zijn relevant voor bredere toepassingen in domeinen waar gelabelde data schaars of moeilijk te verkrijgen is. Hoewel de resultaten veelbelovend zijn, is vervolgonderzoek nodig om de generaliseerbaarheid naar andere types sonar data of real-time toepassingen te evalueren.

# Inhoudsopgave

<b>Lijst van figuren</b>	<b>ix</b>
<b>Lijst van tabellen</b>	<b>x</b>
<b>Lijst van codefragmenten</b>	<b>xi</b>
<b>Woordenlijst</b>	<b>xii</b>
<b>Acroniemen</b>	<b>xv</b>
<b>1 Inleiding</b>	<b>1</b>
1.1 Probleemstelling	1
1.2 Onderzoeksvraag	2
1.3 Onderzoeksdoelstelling	2
1.4 Beschikbaarheid van de code	3
1.5 Opzet van deze bachelorproef	3
<b>2 Stand van zaken</b>	<b>4</b>
2.1 Inleiding	4
2.2 Objectdetectie in sonarafbeeldingen	4
2.2.1 Definitie en gebruik op sonarafbeeldingen	4
Single-shot objectdetectie	5
Two-stage objectdetectie	6
2.2.2 Meten van performantie binnen objectdetectie	6
Intersection over Union (IoU)	7
mean Average Precision (mAP)	7
2.2.3 Typische uitdagingen bij sonarobjectdetectie	7
2.2.4 Overzicht van bestaande technieken	8
Filtertechnieken	9
Thresholding	9
Edge detection	10
YOLO	11
Faster R-CNN	14
SSD	16
2.2.5 Specifieke toepassingen	18

2.3	Semi-supervised learning: principes en technieken	19
2.3.1	Definitie en waarde binnen objectdetectie	19
2.3.2	Veelgebruikte SSL-methoden	21
	Pseudo-labeling	21
	Consistency Regularization	22
	FixMatch	23
	MixMatch	24
2.4	Self-supervised learning: principes en technieken	25
2.4.1	Definitie en verschil met SSL	25
2.4.2	Werking	25
2.4.3	Veelgebruikte Self-SL methoden	26
	SimCLR	27
	MoCo	28
	BYOL	29
2.5	Vergelijking van relevante methoden	31
2.5.1	Gesuperviseerde architecturen	31
2.5.2	Semi-supervised architecturen	32
2.5.3	Self-supervised learning	33
2.6	Overzicht van bestaande datasets en annotatietechnieken	34
2.6.1	Beschikbaarheid van publieke datasets	34
2.6.2	Problemen bij sonardatasets	35
2.6.3	Mogelijke oplossingen	35
	UATD	37
	SSS for Mine Detection	40
	UXO	44
<b>3</b>	<b>Methodologie</b>	<b>48</b>
3.1	Data-acquisitie	48
3.2	Dataverdeling	50
3.3	Modelselectie	50
	3.3.1 Supervised: Faster R-CNN	50
	3.3.2 Semi-supervised: FixMatch	51
	3.3.3 Self-supervised: BYOL pre-training	52
3.4	Resultaten en evaluatie	53
<b>4</b>	<b>Experimenten</b>	<b>55</b>
4.1	Experimentele setup	55
	4.1.1 Deep learning frameworks en tools	55
	TensorFlow	55
	PyTorch	56
	4.1.2 Hardware en compute resources	56
	CPU / GPU / TPU	57

	Toegang tot resources . . . . .	57
4.1.3	Optimalisatietechnieken . . . . .	58
	Early stopping . . . . .	58
	Mixed precision training . . . . .	59
4.2	Baseline: supervised Faster R-CNN . . . . .	60
4.3	Semi-supervised model (FixMatch) . . . . .	62
4.4	Self-supervised model (BYOL) . . . . .	64
	4.4.1 Pre-trainingsfase . . . . .	64
	4.4.2 Downstream objectdetectietaak . . . . .	64
<b>5</b>	<b>Resultaten en evaluatie</b>	<b>66</b>
5.1	Resultatenvergelijking: supervised vs SSL vs Self-SL . . . . .	66
5.2	Kwantitatieve analyse . . . . .	67
5.3	Kwalitatieve analyse . . . . .	69
<b>6</b>	<b>Conclusie</b>	<b>73</b>
6.1	Samenvatting van bevindingen . . . . .	73
6.2	Reflectie op de methodologie . . . . .	74
6.3	Voorstellen voor verder onderzoek . . . . .	75
<b>A</b>	<b>Onderzoeksvoorstel</b>	<b>77</b>
A.1	Inleiding . . . . .	78
A.2	Literatuurstudie . . . . .	78
	A.2.1 Gesuperviseerde Objectdetectie . . . . .	79
	A.2.2 Semi-supervised learning . . . . .	79
	A.2.3 Self-supervised learning . . . . .	80
	A.2.4 Optimalisatie van sonardata . . . . .	81
	A.2.5 Formaat van de data . . . . .	81
	A.2.6 Uitdagingen & opportuniteiten . . . . .	81
	A.2.7 Conclusie . . . . .	82
A.3	Methodologie . . . . .	82
	A.3.1 Fase 1 . . . . .	82
	A.3.2 Fase 2 . . . . .	82
	A.3.3 Fase 3 . . . . .	83
	A.3.4 Fase 4 . . . . .	83
	A.3.5 Fase 5 . . . . .	83
	A.3.6 Fase 6 . . . . .	84
A.4	Verwacht resultaat, conclusie . . . . .	84
	<b>Bibliografie</b>	<b>85</b>

# Lijst van figuren

2.1	Voorstelling van IoU. . . . .	7
2.2	Afbeelding voor en na Otsu's thresholding . . . . .	10
2.3	Originele YOLO-architectuur. . . . .	12
2.4	Faster R-CNN-architectuur. . . . .	14
2.5	Vergelijking YOLO- en SSD-architectuur. . . . .	17
2.6	UATD Objecten. . . . .	37
2.7	UATD afbeelding met bounding boxes. . . . .	38
2.8	Voorbeeld van een bitmap. . . . .	39
2.9	SSS for Mine Detection-afbeelding met bounding boxes. . . . .	41
2.10	Aantal objecten per jaar in SSS for Mine Detection. . . . .	41
2.11	Structuur van een bounding box. . . . .	43
2.12	Voorbeeld van sonarbeelden & afbeeldingen in de UXO-dataset . . . . .	44
2.13	Afbeelding van de ARIS Explorer 3000 . . . . .	45
2.14	Setup waarmee de UXO-dataset gemaakt is . . . . .	45
2.15	UXO-setup in actie . . . . .	47
4.1	Cosine annealing LR . . . . .	62
5.1	mAP voor Faster R-CNN met verschillende data-splits. . . . .	68
5.2	Vergelijking van mAP tussen semi- en self-supervised modellen. . . . .	69
5.3	Voorspellingen door supervised modellen. . . . .	70
5.4	Voorspellingen door semi- en self-supervised modellen. . . . .	71
5.5	Correcte voorspellingen van BYOL 5% . . . . .	72

# Lijst van tabellen

2.1	Vergelijking supervised modellen . . . . .	32
2.2	Vergelijking semi-supervised modellen . . . . .	33
2.3	Vergelijking self-supervised modellen . . . . .	34
2.4	Mogelijke datasets . . . . .	36
2.5	Datasets binnen UATD . . . . .	38
2.6	YOLO-annotatie in getabelleerde vorm . . . . .	42
3.1	Aantal objecten per afbeelding in SSS for Mine Data . . . . .	49
5.1	Overzicht van mAP-score van verschillende modellen. . . . .	67

# Lijst van codefragmenten

2.1	PASCAL VOC-annotatie . . . . .	40
2.2	YOLO-annotatie . . . . .	42
4.1	Variabele warm-up logica . . . . .	61

# Woordenlijst

## Backpropagation

Algoritme om parameters in neurale netwerken te updaten en het model zo te trainen. Het werkt door de fout tussen de voorspelde en werkelijke output terug te propageren door het netwerk, waarbij de gradiënt van de fout ten opzichte van de gewichten wordt berekend. Deze informatie wordt vervolgens gebruikt om de gewichten aan te passen, zodat het netwerk beter presteert bij toekomstige voorspellingen. (Géron, 2023). [xiii](#), [30](#), [56](#), [60](#)

## Batch

Groep van samples die gebruikt worden in één trainingsstap. (Géron, 2023). [27](#), [44](#)

## Batch size

Het aantal samples die gebruikt worden in één trainingsstap. (Géron, 2023). [33](#), [34](#), [44](#), [52](#), [53](#)

## Batch normalisatie

Techniek in deep learning die de invoer van elke laag normaliseert (per batch) waardoor de training minder gevoelig wordt voor initiële waarden van de gewichten en [learning rate](#). (Géron, 2023). [13](#)

## Blindganger

Explosief dat niet is afgegaan. [36](#), [44](#), [45](#), [46](#), [49](#)

## Bounding box

Rechthoek om een object op een afbeelding te identificeren en te lokaliseren. [4](#), [7](#), [12](#), [13](#), [15](#), [16](#), [31](#), [38](#), [41](#), [43](#), [46](#), [48](#), [56](#), [63](#), [70](#), [71](#)

## Buffer underflow

Numerieke fout die ontstaat wanneer een getal (bv. een gradiënt) zo klein is dat het niet nauwkeurig kan worden opgeslagen in float16-formaat en dus afgerond wordt naar nul. Dit heeft als gevolg een verlies van precisie en een slecht of instabiel leerproces. (Micikevicius e.a., 2017). [60](#)

## Confidence score

De zekerheid van het model dat de voorspelling juist is. (Géron, 2023). [12](#), [63](#), [70](#), [71](#)

## Gradient descent

Wiskundig optimalisatie-algoritme dat stap voor stap het minimum van een functie zoekt door telkens in de richting van de grootste daling (de negatieve gradiënt) te bewegen. Dit algoritme gebruikt de gradiënten van de **loss-functie** die door het **backpropagation**-algoritme berekend zijn om de daadwerkelijke parameters aan te passen, zodat de fout afneemt. (Géron, 2023). 58

## Learning rate

Hyperparameter die bepaalt hoe snel het model convergeert richting het minimum van de **loss-functie**. (Géron, 2023). xii, 44, 61, 62, 63, 64

## Loss-functie

Wiskundige functie die de afwijking meet tussen de voorspellingen van een model en de werkelijke waarden. Het doel is om deze afwijking te minimaliseren tijdens het trainen van het model, zodat de nauwkeurigheid van de voorspellingen verbetert. Hoe lager de waarde van de loss-functie, hoe beter het model presteert. (Géron, 2023). xiii, 13, 22, 23, 25, 27, 28, 30, 56, 64

## Mini-batch

Kleine, meer behapbare subset van een batch om het geheugengebruik tijdens training te optimaliseren. (Géron, 2023). 28, 44

## Non-Maximum Supression

Een techniek in computervisie die wordt gebruikt om overlappende detecties te filteren en alleen de meest waarschijnlijke objectlocaties te behouden. Het selecteert de detectie met de hoogste score en onderdrukt omliggende detecties met een hoge overlap (op basis van de **Intersection over Union (IoU)**-score), waardoor dubbele detecties worden verminderd en de nauwkeurigheid van objectdetectie-algoritmes wordt verbeterd. (Géron, 2023). xvi

## Overfitting

Een fenomeen in machine learning en deep learning waarbij het model de trainingsdata te goed leert – inclusief ruis en outliers – in plaats van te generaliseren. Dit zorgt ervoor dat het model excellent presteert op de trainingsdata, maar zeer slecht op ongeziene data.. 48, 58, 64, 73

## Portaalkraan

Een (meestal verrijdbaar) hijswerktuig, opgebouwd uit een portaal waarop een kraan gemonteerd is die verplaatsbaar is langs de horizontale draagbalk. Meestal wordt deze kraan gebruikt op kaden om o.a. containers uit te laden.. 45, 46

## Precision

Een metriek om de accuraatheid van de positieve voorspellingen van het model te meten. Een hoge precision betekent dat het model een lage hoeveelheid *false-positives* voorspelt. Wanneer dit model iets als positief voorspelt, is de kans groot dat dit correct is.

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

(Géron, 2023). 7, 10, 44

## Recall

Een metriek om te meten hoe goed het model alle relevante positieve datapunten kan voorspellen. Een hoge recall betekent dat het model heel goed is in het correct voorspellen van alle positieve datapunten, zelfs als dit betekent dat het meer *false-positives* voorspelt.

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

(Géron, 2023). 7, 44

## Thermocline

De overgang tussen twee lagen water (of lucht) met verschillende temperatuur en dichtheid, in bijvoorbeeld meren of oceanen.. 8

# Acroniemen

## **AMP**

Automatic Mixed Precision. [60](#), [63](#), [64](#)

## **AR**

Augmented Reality. [11](#), [18](#)

## **ASIC**

Application Specific Integrated Circuit. [57](#)

## **AUV**

Autonomous Underwater Vehicle. [14](#), [35](#), [36](#), [37](#), [40](#)

## **BYOL**

Bootstrap Your Own Latent. [2](#), [26](#), [29](#), [30](#), [31](#), [33](#), [34](#), [52](#), [53](#), [64](#), [65](#), [66](#), [67](#), [68](#), [69](#), [70](#), [71](#), [72](#), [73](#), [74](#), [75](#), [76](#)

## **CAS**

Continuous Active Sonar. [9](#)

## **CNN**

Convolutioneel Neuraal Netwerk. [5](#), [12](#), [16](#), [27](#), [56](#)

## **COCO**

Common Objects in Context. [44](#), [56](#)

## **CPU**

Central Processing Unit. [57](#)

## **DFKI**

Deutsches Forschungszentrum für Künstliche Intelligenz. [44](#)

## **DINO**

Self-Distillation with No Labels. [75](#)

## **DMS 3**

Destacamento de Mergulhadores Sapadores. [40](#)

**EMA**

Exponential Moving Average. 30

**FAIR**

Facebook AI Research. 56

**GAN**

Generative Adversarial Networks. 21

**GPU**

Graphics Processing Unit. 57, 58, 60, 75

**IoU**

Intersection over Union. xiii, 6, 7, 44

**KL**

Kullback-Leibler. 22

**mAP**

Mean Average Precision. 5, 6, 7, 44, 53, 62, 63, 67, 68, 69, 73, 74

**MFLS**

Multibeam Forward-Looking Sonar. 37

**MILCO**

MIne-Like COntact. 41, 43

**MoCo**

Momentum Contrast. 26, 28, 29, 33, 34, 52, 53, 75

**MSE**

Mean Squared Error. 22, 30

**NLP**

Natural Language Processing. 21, 22

**NMS**

Non-Maximum Supression. 12, 15, 17, 71

**NOMBO**

NOOn-Mine-like BOttom Object. 41, 43

**NT-Xent**

Normalized Temperature-scaled Cross Entropy. 27

**PTU**

Pan-Tilt Unit. 46

**R-CNN**

Region-based Convolutional Neural Network. 5, 6, 14, 15, 16, 19, 26, 31, 32, 50, 51, 52, 53, 55, 60, 61, 62, 64, 65, 67, 68, 69, 70, 71, 72, 73, 74

**RoI**

Region of Interest. 15, 16

**ROV**

Remotely Operated Vehicle. 35, 36

**RPN**

Region Proposal Network. 15, 16, 31, 51

**Self-SL**

Self-Supervised Learning. 2, 3, 4, 25, 26, 27, 28, 29, 31, 33, 34, 48, 50, 51, 52, 53, 55, 59, 60, 61, 64, 66, 67, 68, 69, 70, 72, 73, 74, 75, 76

**SGD**

Stochastic Gradient Descent. 63, 65

**SimCLR**

Simple Framework for Contrastive Learning of Visual Representations. 2, 26, 27, 31, 33, 34, 52, 53, 75

**SSD**

Single Shot Multibox Detector. 5, 6, 16, 17, 18, 19, 25, 31, 32, 50, 51

**SSL**

Semi-Supervised Learning. 2, 3, 4, 19, 20, 21, 22, 23, 24, 25, 26, 31, 32, 33, 48, 50, 51, 52, 55, 59, 60, 61, 62, 66, 67, 68, 69, 70, 72, 73, 74, 75, 76

**SSS**

Side-Scan Sonar. 40, 41, 48, 49

**TPU**

Tensor Processing Unit. 56, 57

**UDA**

Unsupervised Data Augmentation. 75

**UXO**

Unexploded Ordnance. 44

**VAE**

Variational Autoencoder. 21

**VOC**

Visual Object Classes. 5, 39, 56

**YOLO**

You Only Look Once. 5, 6, 11, 12, 13, 14, 17, 18, 19, 31, 32, 42, 44, 50, 51

# 1

## Inleiding

Objectdetectie heeft de afgelopen jaren enorme vooruitgang geboekt dankzij de opkomst van deep learning en de beschikbaarheid van grote, gelabelde datasets. In domeinen zoals computervisie, waar overvloedige trainingsdata gemakkelijk toegankelijk is, hebben deze technieken indrukwekkende prestaties bereikt. Echter beschikken niet alle vakgebieden over dergelijke datasets. In gespecialiseerde domeinen, zoals sonarbeeldanalyse, is gelabelde data schaars, wat het trainen van nauwkeurige detectiemodellen bemoeilijkt. Dit onderzoek richt zich op het verkennen van alternatieve leermethoden die deze afhankelijkheid van handmatige annotatie kunnen verminderen, zonder in te boeten op de prestaties van het model.

### 1.1. Probleemstelling

Om een model te trainen dat met hoge precisie objecten in afbeeldingen kan herkennen en aanduiden, is een grote hoeveelheid gelabelde data nodig. Dit betekent dat naast de afbeeldingen zelf ook informatie over de positie van het object op de afbeelding beschikbaar moet zijn. Na de training is het de bedoeling dat het model deze informatie kan voorspellen op ongekende afbeeldingen. Sinds de opkomst van objectdetectie binnen het veld van machine learning zijn verschillende datasets openbaar beschikbaar gesteld die vrij gebruikt mogen worden om een dergelijk model te trainen. Dit geldt echter niet voor sonardata, om meerdere redenen.

Allereerst is objectdetectie op sonarbeelden een nicheprobleem. Hierdoor beschikken slechts weinig mensen over de kennis en expertise om zo'n dataset samen te stellen en, nog belangrijker, correct te annoteren. Daarnaast is er het – misschien nog grotere – probleem van de data zelf. Dit type gegevens kan niet eenvoudig met

een camera worden verzameld; er is een gespecialiseerde sonarinstallatie voor nodig. Voor bedrijven is de aanschaf van zo'n installatie vaak weinig rendabel. Wel worden dergelijke systemen gebruikt voor militaire doeleinden, maar de data die hieruit voortkomt, is om veiligheidsredenen vrijwel altijd geclassificeerd.

De probleemstelling is dus tweeledig: er is weinig gelabelde sonardata voor objectdetectie beschikbaar, en het annoteren van een dergelijke dataset is moeilijk, tijdrovend en kostbaar.

## 1.2. Onderzoeksvraag

Om kosten en tijd te besparen, zou het dus ideaal zijn als er zo min mogelijk annotatie van de dataset nodig is. Bij supervised learning is dit echter nagenoeg onmogelijk, aangezien het model juist getraind wordt op basis van het verband tussen de afbeelding en de annotatie. Er bestaan echter veelbelovende alternatieven, zoals [Semi-Supervised Learning \(SSL\)](#) en [Self-Supervised Learning \(Self-SL\)](#), om dit probleem te overbruggen. Deze technieken maken gebruik van ongesuperviseerde data om hun performantie te verbeteren en beperken zo de afhankelijkheid van gelabelde data. Moderne [SSL](#) en [Self-SL](#) methoden hebben indrukwekkende resultaten laten zien in domeinen zoals computer vision, maar hun toepassing op domeinspecifieke datasets, zoals sonar, is nog relatief onbekend terrein.

De hoofdvraag van dit onderzoek is daarom: Op welke manieren kan het gebruik van [SSL](#) of [Self-SL](#) het labelproces versnellen zonder een significant verlies in nauwkeurigheid?

## 1.3. Onderzoeksdoelstelling

Het onderzoek verwacht aan te tonen dat [SSL](#) en [Self-SL](#) effectief kunnen worden ingezet om het labelproces bij sonarobjectdetectie aanzienlijk te versnellen. Door technieken zoals [Simple Framework for Contrastive Learning of Visual Representations \(SimCLR\)](#), [Bootstrap Your Own Latent \(BYOL\)](#), Pseudo-Labeling en FixMatch te gebruiken, wordt verwacht dat de performantie van het model sterk verbeterd door ongesuperviseerde sonardata, waardoor de behoefte aan grootschalige gelabelde datasets afneemt. Daarnaast zal een analyse inzicht geven in de minimale hoeveelheid gelabelde data die nodig is om vergelijkbare of betere prestaties te behalen dan met volledig supervised-learning methoden.

Dit resulteert in een efficiëntere en kosteneffectieve aanpak voor objectdetectie in sonarbeelden, zonder verlies van nauwkeurigheid, en biedt een waardevolle methodologie voor verdere toepassingen in domeinen waar gelabelde data schaars is.

## 1.4. Beschikbaarheid van de code

Alle code die gebruikt werd voor de experimenten en implementatie in deze bachelorproef is publiek beschikbaar via een GitHub-repository. De repository bevat de volledige implementatie van de gebruikte modellen, inclusief de baseline, het *SSL*-model en het *Self-SL*-model. Daarnaast zijn ook de scripts voor training, evaluatie en visualisatie van de resultaten opgenomen. De code is modulair opgezet en voorzien van duidelijke documentatie, zodat reproduceerbaarheid en verdere uitbreidingen mogelijk zijn. De repository is te vinden op: [github.com/Yoran-Gyselen/bsc-thesis-sonar-object-detection](https://github.com/Yoran-Gyselen/bsc-thesis-sonar-object-detection). Neem voor vragen over dit onderzoek contact op met de auteur via het e-mailadres [yoran@gyselen.be](mailto:yoran@gyselen.be).

## 1.5. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4 wordt het eigenlijke onderzoek uitgevoerd. In deze fase wordt het experimentele kader verduidelijkt en zal er dieper ingegaan worden op de effectieve uitvoering van de verschillende experimenten.

In Hoofdstuk 5 worden de getrainde modellen geëvalueerd op basis van verschillende criteria. Er wordt een vergelijking gemaakt tussen – onder andere – de performantie van elk model en er wordt bepaald welk model de beste is. Daarnaast wordt de praktische toepassing van de verschillende modellen geëvalueerd. Ook zullen enkele experts in sonaranalyse de bruikbaarheid van de resultaten beoordelen en aanbevelingen geven voor verdere verbeteringen.

In Hoofdstuk 6, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

# 2

## Stand van zaken

### 2.1. Inleiding

Objectdetectie in domeinspecifieke contexten zoals sonarbeeldvorming wordt vaak gehinderd door een gebrek aan gelabelde data. Traditioneel vereisen gesuperviseerde modellen grote hoeveelheden handmatig gelabelde gegevens om effectieve detectie en classificatie te leren. [SSL](#) en [Self-SL](#) bieden echter veelbelovende alternatieven door gebruik te maken van grote hoeveelheden ongesuperviseerde data. Deze literatuurstudie bespreekt de huidige technieken en hun toepassing, met een specifieke focus op de unieke uitdagingen van sonardata.

### 2.2. Objectdetectie in sonarafbeeldingen

#### 2.2.1. Definitie en gebruik op sonarafbeeldingen

Objectdetectie is een tak binnen het domein van computer vision dat gericht is op het identificeren en lokaliseren van objecten binnen beelddata (zoals foto's en video's). Dit wordt gebruikt in verschillende domeinen, zoals beveiligingssystemen (bv. om inbrekers te detecteren) of de medische wereld (bv. om tumoren op te sporen). Door de jaren heen is objectdetectie aanzienlijk geëvolueerd dankzij de vooruitgang in deep learning en de grote beschikbaarheid van datasets met beeldmateriaal. (He e.a., [2016](#))

Objectdetectie combineert twee belangrijke zaken in computer vision: objectlokalisatie en objectclassificatie. Objectlokalisatie bepaalt de positie van objecten, meestal in de vorm van [bounding boxes](#) (Tompson e.a., [2015](#)), terwijl objectclassificatie bepaalt tot welke categorie een gedetecteerd object behoort. Samen geeft dit de mogelijkheid tot het herkennen van verschillende soorten objecten op één afbeelding.

Objectdetectie heeft vele toepassingen, ook in domeinen die misschien niet zo voor de hand liggend zijn. Één van deze specialisaties binnen de – algemene – objectdetectie is objectdetectie op sonardata. Dit domein is de laatste jaren erg gegroeid, vooral onder invloed van buitenlandse dreigingen. Zo wordt sonarobjectdetectie gebruikt voor het opsporen van mijnen in zee om ze later onschadelijk te kunnen maken. Naast detectie van mijnen wordt de techniek ook gebruikt voor verschillende soorten onderzoeken, zoals archeologisch en maritiem onderzoek. Bij deze verschillende toepassingen wordt natuurlijk telkens een kleine variatie op deze techniek gebruikt om telkens andere dingen op te sporen. (W. Wang e.a., 2024)

Traditioneel worden supervised learning methoden gebruikt voor objectdetectie. Voorbeelden van populaire architecturen binnen dit domein zijn onder andere Faster Region-based Convolutional Neural Network (R-CNN), You Only Look Once (YOLO) en Single Shot Multibox Detector (SSD). (Redmon e.a., 2016) Omdat dit supervised learning modellen zijn, presteren ze uitstekend bij voldoende gelabelde data. De annotatiekosten en tijdsinvestering vormen echter een grote belemmering, vooral bij complexe datasets zoals sonar. Sonardata vereist namelijk gespecialiseerde kennis voor het labelen, wat de annotatie nog uitdagender maakt. (Long e.a., 2015)

Deze supervised objectdetectietechnieken vallen onder te verdelen in grofweg twee grote categorieën. Enerzijds zijn er de zogenaamde *single-shot detectors*, anderzijds heb je de *two-stage detectors*. Deze categorieën zijn gebaseerd op hoeveel keer de afbeelding door het netwerk gaat. Bij single-shot detectors gaat de afbeelding slechts één keer door het netwerk, bij two-stage detectors – logischerwijs – twee keer. (Carranza-García e.a., 2020)

Één van de eerste succesvolle toepassingen van deep learning binnen het domein van objectdetectie gebeurde in een bekend artikel van Girshick e.a. (2013). In dit artikel stelden de auteurs de R-CNN-architectuur voor. Deze veelbelovende architectuur behaalde een Mean Average Precision (mAP) van 30% meer dan de vorige topscore op een bekende publieke dataset (VOC 2012).

### **Single-shot objectdetectie**

Zoals hierboven vermeld, is een single-shot detector een model waarbij de afbeelding slechts één keer door het netwerk gaat. Dit gebeurt door een Convolutioneel Neuraal Netwerk (CNN) te gebruiken dat zowel objectlocaties als bijbehorende classificaties voorspelt in één enkele *pass*. Het voordeel van dit soort architectuur is dat ze zeer resource-efficiënt is, aangezien elke afbeelding slechts één keer behandeld wordt. Omwille van de efficiëntie, is deze architectuur dus uitermate geschikt voor real-time toepassingen zoals autonome voertuigen en bewakingssystemen. Een nadeel is echter dat het model niet altijd even precies is, aangezien het zo-

wel locaties als klassen in één pass moet voorspellen. Vooral bij het detecteren van kleine objecten geeft dit een probleem. Om dit – toch gedeeltelijk – op te lossen, wordt soms gebruik gemaakt van meerdere schaalniveaus om objecten van verschillende groottes te detecteren, wat bijdraagt aan de robuustheid en precisie. (Carranza-García e.a., 2020)

Zowel YOLO als SSD zijn single-shot detectors. Deze architecturen worden verder besproken in latere secties hieronder.

### Two-stage objectdetectie

Zoals de naam doet vermoeden, is two-stage objectdetectie een type objectdetectie waarbij de afbeelding twee keer door het netwerk gaat. Het resultaat is nog steeds een voorspelling van locaties en klassen, maar in plaats van in één keer – zoals bij single-shot objectdetectie – gebeurt de voorspelling nu in twee afzonderlijke stappen. Het resultaat van de eerste pass is een set van *proposals* – voorstellen – en mogelijke locaties van objecten. Daarna zorgt de tweede pass voor een verfijning van deze voorstellen. Dit zorgt dan ook voor de uiteindelijke voorspellingen. Het voordeel van dit type objectdetectie is dat het veel preciezer is dan single-shot objectdetectie, aangezien de afbeelding twee keer geanalyseerd wordt. Het nadeel is dat deze approach veel resource-intensiever is. Het is dus zaak om een afweging te maken tussen de precisie van de voorspellingen en het verbruik van resources. (Carranza-García e.a., 2020)

Over het algemeen wordt voor real-time applicaties single-shot objectdetectie gebruikt en voor cases waar de voorspellingen heel accuraat moeten zijn, wordt two-stage objectdetectie gebruikt. Een voorbeeld van een two-stage detector is de R-CNN-architectuur van Girshick e.a. (2013). Ook alle latere architecturen die hierop gebaseerd zijn, zijn two-stage detectors. Voorbeelden daarvan zijn Fast R-CNN, Faster R-CNN en Mask R-CNN. (Ren e.a., 2015)


#### 2.2.2. Meten van performantie binnen objectdetectie

Gedurende dit onderzoek zal er gewerkt worden met verschillende soorten modellen en architecturen met elk hun sterktes en hun zwaktes. Het is de bedoeling dat al deze modellen vergeleken kunnen worden met elkaar om zo het best presterende model of de best presterende modellen te kunnen selecteren. Om dit te doen zijn er standaard metrieken nodig. Deze kunnen later (tijdens de trainingsfase) dan ook dienen als tussentijdse evaluatiemetriek om te zien hoe het model evolueert. Er zijn enorm veel verschillende metrieken om de performantie van objectdetectiemodellen te meten, maar twee van de bekendste en meest gebruikte zijn IoU en mAP.

### Intersection over Union (IoU)

IoU is een heel bekende metriek om de accuraatheid van de lokalisatie binnen objectdetectiemodellen te berekenen. Om de IoU te berekenen wordt gebruikt gemaakt van de echte **bounding box** en de voorspelde **bounding box**. Eerst wordt de overlappende oppervlakte (de doorsnede of *intersection* in het Engels) van de twee **bounding boxes** berekend. Daarna wordt de totale oppervlakte (de unie of *union* in het engels) van de twee **bounding boxes** berekend.

Door de doorsnede te delen door de unie krijgt men een verhouding van de overlappende oppervlakte tot de totale oppervlakte. Dit geeft een goede indicatie van hoe dicht de voorspelde **bounding box** bij de echte **bounding box** ligt. Een lagere IoU-score duidt op een betere prestatie, aangezien de voorspelde **bounding box** niet dan weinig afwijkt van de echte **bounding box**. (Rezatofighi e.a., 2019)

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


**Figuur 2.1:** Formule en visuele voorstelling van de **Intersection over Union (IoU)**. (Rosebrock, 2016)

### mean Average Precision (mAP)

Een andere – zeer bekende – metriek is de **Mean Average Precision (mAP)**. Ze beoordeelt de nauwkeurigheid van een model op basis van zowel **precision** als **recall**. Het wordt berekend door de gemiddelde precisie (of *average precision*, AP) over alle klassen en IoU drempels te bepalen. AP wordt verkregen door de oppervlakte onder de precisie-recallcurve van een specifieke klasse te berekenen door hem te integreren, en mAP is vervolgens het gemiddelde van deze waarden over alle klassen. Een hogere mAP-score duidt op betere prestaties van een objectdetectiemodel, omdat het aangeeft hoe goed het model objecten correct detecteert en classificeert. (B. Wang, 2022)

### 2.2.3. Typische uitdagingen bij sonarobjectdetectie

Objectdetectie op sonarbeelden die gemaakt zijn onderwater wordt geconfronteerd met verschillende uitdagingen die de nauwkeurigheid en betrouwbaarheid van detecties beïnvloeden.

Ruis is één van de voornaamste obstakels bij sonarobjectdetectie. Onderwateromgevingen zijn inherent lawaaiër door factoren zoals luchtbellen, **thermoclines** en biologische organismen, wat leidt tot significante ruis in sonarbeelden. Deze ruis bemoeilijkt het onderscheiden van echte objecten van artefacten typisch aan sonarbeelden, waardoor de betrouwbaarheid van de detecties afneemt. (Aubard, Madureira e.a., 2024)

Een andere uitdaging is de lage resolutie van sonarbeelden. In vergelijking met optische sensoren leveren sonars vaak beelden met beperkte detailweergave, wat de identificatie en classificatie van objecten moeilijker maakt. Deze beperking is vooral problematisch bij het detecteren van kleine of complexe objecten, waar detailniveau essentieel is voor nauwkeurige herkenning. (S. Lee e.a., 2018)

Daarnaast zorgen variërende omstandigheden in de onderwateromgeving voor extra complicaties. Factoren zoals veranderende waterdieptes, temperatuurverschillen, stromingen en de aanwezigheid van zwevende deeltjes kunnen de prestaties van sonarsystemen beïnvloeden. Deze dynamische omstandigheden kunnen leiden tot variaties in signaalsterkte en -kwaliteit, wat de consistentie van objectdetectie bemoeilijkt. (Valdenegro-Toro, 2019)

Het overwinnen van deze uitdagingen vereist geavanceerde signaalverwerkings technieken en robuuste algoritmen die kunnen omgaan met ruis, lage resolutie en variabele omgevingsfactoren. Door voortdurende technologische innovaties en onderzoek kunnen de prestaties van sonarobjectdetectiesystemen worden verbeterd, wat leidt tot betrouwbaardere toepassingen in onderwateromgevingen.

#### **2.2.4. Overzicht van bestaande technieken**

Grofweg zijn er twee stromingen van objectdetectie op sonarafbeeldingen. Enerzijds zijn er de klassieke methoden en anderzijds zijn er de moderne deep learning-technieken. De klassieke methoden werden vooral gebruikt in een tijd waar grote, complexe neurale netwerken trainen onmogelijk was bij gebrek aan voldoende computerkracht, maar worden de dag van vandaag nog altijd gebruikt als pre-processing technieken voor de datasets waarmee de moderne neurale netwerken getraind worden. Deze klassieke methoden berusten enkel op statistische technieken om zo objecten in afbeeldingen te proberen detecteren. Specifiek zijn deze vooral gericht op het verbeteren van beeldkwaliteit en het onderscheiden van objecten van de achtergrond.

### Filtertechnieken

Een voorbeeld van een klassieke methode zijn filtertechnieken. Deze worden toegepast om ruis in sonarafbeeldingen te verminderen en de beeldkwaliteit te verbeteren. Er bestaan immens veel verschillende soorten filters die elk geoptimaliseerd voor een specifiek doel. Een veelgebruikte filtermethode is het gebruik van adaptieve filters die zich aanpassen aan de lokale kenmerken van het beeld. Een voorbeeld hiervan is te vinden in een artikel van Aridgides e.a. (1995). Merk op dat dit inderdaad een relatief oude publicatie is, wat aantoont dat deze technieken al gebruikt werden toen deep learning-gebaseerde objectdetectie niet mogelijk was.

In dit artikel introduceren de auteurs een adaptieve filtertechniek die ontwikkeld is om mijnachtige doelen te onderscheiden van achtergrondruis in sonarbeelden. De filter onderdrukt achtergrondruis terwijl het de target behoudt. De procedure omvat vier stappen: het berekenen van een genormaliseerde gemiddelde target, het bepalen van de covariantiematrix van de achtergrondruis, het oplossen van normale vergelijkingen om een adaptief filter te verkrijgen en het toepassen van een 2D-filter op de gegevens. Dit algoritme bewijst dat, hoewel er geen gebruik gemaakt wordt van deep-learningtechnieken, ze toch complex kan zijn. De techniek heeft in verschillende testen prestaties geleverd die vergelijkbaar zijn met die van een ervaren sonaroperator.

Adaptieve filters worden ook vandaag de dag nog gebruikt, wat aangetoond wordt door een paper van Lourey (2017). Hierin wordt ook een filtertechniek beschreven die toegepast kan worden op **Continuous Active Sonar (CAS)** om interferentie van de directe transmissie en de echo van de target van elkaar te onderscheiden. Deze methode kan als effectieve pre-processing techniek gebruikt worden voor trainingsdata.

### Thresholding

Naast filtering bestaan er nog andere klassieke methoden. Een *straightforward*-aanpak is een simpele *threshold*. Thresholding is een techniek waarbij pixelwaarden worden vergeleken met een bepaalde drempelwaarde om objecten van de achtergrond te scheiden. Een klassieke benadering is de Otsu-methode, die de interklassevariantie minimaliseert om een optimale drempelwaarde te bepalen. Deze methode wordt beschreven in een artikel van Otsu (1979).



(a) Afbeelding voor Otsu's thresholding.  
(<http://www.freepotos.lu/>, 2010)



(b) Afbeelding na Otsu's thresholding. (Pikez33, 2010)

**Figuur 2.2:** Afbeelding voor en na Otsu's thresholding

Hoewel deze techniek oorspronkelijk is ontwikkeld voor visuele beelden, is deze ook toegepast op sonarafbeeldingen, zoals besproken in verschillende artikels, waaronder dat van Yuan e.a. (2016) en dat van Dimitrova-Grekow e.a. (2017). Ondanks zijn simpliciteit kan deze techniek aanzienlijke verbeteringen teweegbrengen. Dit wordt onder andere aangehaald in een paper van Komari Alaie en Farsi (2018). Deze paper onderzoekt objectdetectie met passieve sonar in de Perzische Golf. Aangezien deze binnenzee ondiep is, is er sprake van een hoge hoeveelheid fouten tijdens de detectie. Een bepaald soort adaptieve thresholding-techniek kon de **precision** van hun objectdetectiemodel met 24% verbeteren.

### Edge detection

Edge detection is een andere klassieke techniek die wordt gebruikt om de contouren van objecten in sonarafbeeldingen te identificeren. (Torre & Poggio, 1986) Een bekende methode is de Canny edge detector, die randen detecteert door het maximaliseren van de gradiëntgrootte. Deze techniek komt als beste uit de vergelijkende studie van Awalludin e.a. (2022).

De Canny edge detector werkt in meerdere stappen om nauwkeurige en robuuste contourdetectie te realiseren. De eerste stap is Gaussian blurring, waarbij het beeld wordt vervaagd om ruis te verminderen en kleine details die geen significante randen vormen te onderdrukken. Vervolgens wordt de gradiënt van het beeld berekend met behulp van Sobel-operatoren in zowel de horizontale als verticale richting, waardoor de randen worden geaccentueerd. Daarna wordt non-maximum suppression toegepast, waarbij alleen de sterkste randen worden behouden en omliggende pixels met lagere gradiëntwaarden worden onderdrukt. De laatste stap is hysteresis thresholding, waarbij twee drempelwaarden worden gebruikt: pixels met een gradiëntsterkte boven de hoge drempel worden als randen geclassificeerd, terwijl pixels onder de lage drempel worden genegeerd. Pixels met tus-

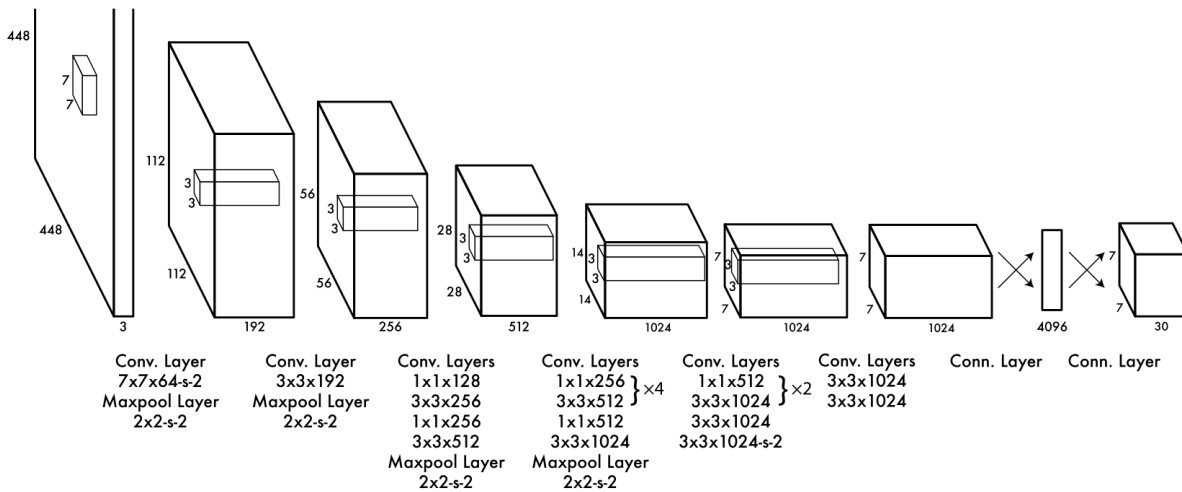
senliggende waarden worden alleen als rand beschouwd als ze verbonden zijn met een sterke randpixel. Dankzij deze gefaseerde aanpak is de Canny-methode effectief in het detecteren van duidelijke randen, zelfs in ruisgevoelige omgevingen zoals sonarafbeeldingen. (Ding & Goshtasby, 2001)

Ook edge detection wordt vandaag de dag nog gebruikt om een grote bijdrage te leveren aan bijvoorbeeld segmentatiemodellen. Het onderzoek van Priyadharsini en Sharmila (2019) gebruikt gespecialiseerde edge detection als pre-processing voor de data naar een objectdetectiemodel gaat.

Deze klassieke technieken vormen de basis voor objectdetectie in sonarafbeeldingen en hebben bijgedragen aan de ontwikkeling van meer geavanceerde methoden. Ze blijven relevant, vooral in situaties waarin resources beperkt zijn of wanneer eenvoud en interpretatie van het model belangrijk zijn. Ze worden tot op de dag van vandaag gebruikt als pre-processing stap, bijvoorbeeld. Doordat computerkracht steeds goedkoper en meer beschikbaar werd, wordt tegenwoordig vaak gekozen voor deep learning-oplossingen voor deze problemen. Er zijn gespecialiseerde architecturen ontwikkeld om objectdetectie uit te voeren. Hieronder worden er enkele besproken.

### YOLO

**You Only Look Once (YOLO)** is een deep learning-gebaseerde architectuur voor objectdetectie dat bekend staat om zijn snelheid en efficiëntie. Het werd voor het eerst geïntroduceerd in een artikel van Redmon e.a. (2016) en is sindsdien één van de populairste algoritmes in computervisie. In tegenstelling tot traditionele detectiemethoden, waar objecten in meerdere stappen geanalyseerd worden, verwerkt YOLO een afbeelding in één enkele *pass* van het neurale netwerk. Dit zorgt ervoor dat real-time objectdetectie mogelijk is, waardoor het bijzonder geschikt is voor toepassingen zoals autonome voertuigen, videobewaking en **Augmented Reality (AR)**.



**Figuur 2.3:** Schematische voorstelling van de originele YOLO-architectuur. (Redmon e.a., 2016)

YOLO gebruikt een CNN om objecten direct te lokaliseren en classificeren, wat bijdraagt aan de hoge nauwkeurigheid en snelheid van het model. De eerste versie van YOLO gebruikt ImageNet om de eerste 20 convolutionele lagen te pre-trainen. Het model wordt daarna omgezet om detectie uit te voeren, aangezien de combinatie van convolutionele lagen en fully-connected-lagen de performantie verhoogt. (Redmon e.a., 2016) De interne werking van YOLO kan opgesplitst worden in verschillende fasen.

Eerst en vooral wordt de invoerafbeelding opgesplitst in een  $S \times S$  raster (bv.  $7 \times 7$ ). Elke cel van dat raster is daarna verantwoordelijk voor het detecteren van objecten waarvan het midden zich in dat vak bevindt. Voor elke cel voorspelt YOLO meerdere bounding boxes. Zo'n voorspelling van een bounding box bevat telkens 5 parameters (cf. 2.11):

- $x$  en  $y$ : de gecentreerde coördinaten van het object binnen de cel in het raster.
- $w$  en  $h$ : de breedte en hoogte van het object, genormaliseerd ten opzichte van de afbeelding.
- De confidence score: de waarschijnlijkheid dat er daadwerkelijk een object in de box zit.

Naast het voorspellen van de bounding boxes voorspelt het model ook de klasse van het object (bv. auto, hond, persoon, ...) en de confidence score van deze classificatie. Echter zijn er vaak meerdere bounding boxes die hetzelfde object detecteren. Daarom gebruikt YOLO Non-Maximum Suppression (NMS) om overbodige detecties te verwijderen. Ten slotte genereert YOLO een lijst met gedetecteerde objecten, hun locaties en de waarschijnlijkheid van hun klassen. (Diwan e.a., 2022)

YOLO is sneller dan traditionele methoden zoals Faster R-CNN omdat het objectdetectie beschouwt als een enkel regressieprobleem. Dit betekent dat het model direct van ruwe pixels naar detecties gaat, zonder een apart proces voor regiovoorstel en classificatie.

Sinds de introductie van YOLO in de paper van Redmon e.a. (2016) heeft het model aanzienlijke verbeteringen en evoluties doorgemaakt. De oorspronkelijke versie, YOLOv1, legde de basis met een enkelvoudige doorvoer voor objectdetectie, maar had beperkingen in nauwkeurigheid, vooral bij kleine objecten. YOLOv2 (ook wel YOLO9000) werd kort na de introductie van het originele YOLO-model geïntroduceerd in een paper van Redmon en Farhadi (2016). De nieuwe versie werd ontwikkeld om sneller en accurater te zijn dan het originele model. Ook kon deze versie meer verschillende klassen onderscheiden. Daarnaast werd er gebruik gemaakt van een andere *backbone*, namelijk Darknet-19 (wat zelf een variant is van VGGNet). Dit zijn de belangrijkste veranderingen:

- Gebruik van een nieuwe *loss-functie* die beter geschikt is voor objectdetectie.
- Gebruik van *batch normalisatie* om accuraatheid en stabiliteit te verhogen.
- Trainen op zelfde afbeeldingen met een verschillende schaal, hierdoor wordt het model beter in het herkennen van kleine objecten.
- Gebruik van zogenaamde *anchor boxes*: dit zijn vooraf gedefinieerde *bounding boxes* die helpen bij het detecteren van objecten in een afbeelding. Tijdens het trainen leert het model welke van deze *anchor boxes* het beste passen bij de werkelijke objecten in de afbeelding.

YOLOv3 werd geïntroduceerd in een paper van Redmon en Farhadi (2018). Het doel van deze iteratie was opnieuw het verbeteren van de accuraatheid en de snelheid. Dit deden de onderzoekers door opnieuw een andere architectuur te gebruiken als *backbone*. Dit keer gebruikten ze Darknet-53 (een variant van ResNet). Daarnaast werden de *anchor boxes* aangepast zodat ze verschillende vormen en maten hadden (in tegenstelling tot allemaal dezelfde vorm en maat in YOLOv2). Ook werden nog andere technieken toegepast om kleine objecten efficiënter en beter te kunnen detecteren.

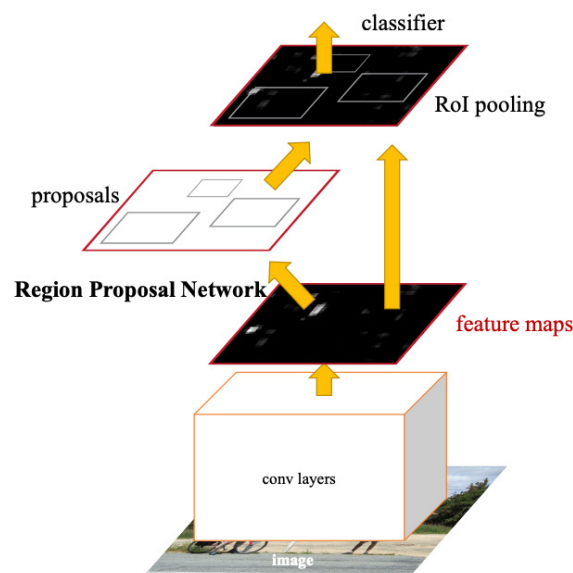
De ontwikkeling van YOLO werd overgenomen door andere mensen, aangezien Joseph Redmond (de originele ontwikkelaar) na YOLOv3 de AI-community verliet. In een paper van Bochkovskiy e.a. (2020) werd YOLOv4 geïntroduceerd. Hierin werd de efficiëntie verder verhoogd door verbeterde activatiefuncties en optimalisatie-technieken. YOLOv5, geïntroduceerd door Ultralytics, richtte zich op gebruiksvriendelijkheid en efficiëntere implementatie. (P. Jiang e.a., 2022) Nieuwere versies, zoals YOLOv7 en YOLOv8, blijven innoveren met hogere detectienauwkeurigheid, ver-

beterde verwerkingstijden en geavanceerdere architecturen, waardoor YOLO één van de meest gebruikte objectdetectiemodellen blijft in real-time toepassingen. (Terven e.a., 2023)

Ook heeft de architectuur veel potentie voor onderwaterobjectdetectie, zoals bij het opsporen van wrakken, onderzeese mijnen en mariene organismen. Dankzij de snelheid en efficiëntie van YOLO kunnen real-time detecties worden uitgevoerd, wat waardevol is voor Autonomous Underwater Vehicles (AUVs) en robots die in onbekende of gevaarlijke omgevingen opereren. Bovendien kunnen verbeterde versies van YOLO, zoals YOLOv5 en YOLOv8, met aangepaste architecturen en pre-processingtechnieken betere resultaten behalen op sonarbeelden. Door de voortdurende ontwikkeling van deep learning en sonarverwerking wordt YOLO steeds vaker ingezet voor geavanceerde onderwaterdetectie en navigatie. (C. Chen e.a., 2023)

### Faster R-CNN

Een significante ontwikkeling in het domein van objectdetectie was de introductie van Faster R-CNN in een artikel van Ren e.a. (2015). Deze architectuur is een state-of-the-art model dat de snelheid en nauwkeurigheid van objectdetectie aanzienlijk heeft verbeterd. Het woord “Faster” in Faster R-CNN impliceert een duidelijke verbetering in snelheid ten opzichte van zijn voorgangers binnen de R-CNN-familie. Het feit dat Faster R-CNN zelfs meer dan tien jaar na zijn introductie nog steeds als referentiepunt wordt gebruikt in veel nieuwe objectdetectiepapers, benadrukt zijn blijvende relevantie en hoge mate van nauwkeurigheid.



**Figuur 2.4:** Schematische voorstelling van de Faster R-CNN-architectuur. (Ren e.a., 2015)

Faster R-CNN is een zogenaamd two-stage detectiemodel, wat inhoudt dat het eerst regio's in een afbeelding identificeert die mogelijk objecten bevatten, en deze vervolgens classificeert en verfijnt. Hoewel dit proces doorgaans minder snel is dan bij single-stage modellen, biedt het een grotere nauwkeurigheid in zowel lokalisatie als classificatie, wat verklaart waarom het model goed presteert in veel benchmarks.

De architectuur bestaat uit vier hoofdcomponenten:

1. Een convolutionele backbone
2. Het Region Proposal Network (RPN)
3. De Region of Interest (RoI) Pooling (of Align) laag
4. De classificatie- en regressieheads

De convolutionele backbone – vaak een gepretraïnd netwerk zoals VGG-16 of ResNet – is verantwoordelijk voor het extraheren van visuele kenmerken uit de invoerafbeelding. De keuze van backbone beïnvloedt zowel nauwkeurigheid als computationele resources: diepere netwerken zoals ResNet kunnen complexere patronen leren, maar vereisen ook meer resources.

Het RPN vormt het hart van Faster R-CNN. Dit volledig convolutionele netwerk genereert potentiële objectlocaties direct vanuit de feature map. Het doet dit door op elke locatie meerdere zogenaamde *anchors* te plaatsen: vooraf gedefinieerde **bounding boxes** met diverse schalen en beeldverhoudingen. Voor elk anker voorspelt het RPN een objectness score en voert het **bounding box** regressie uit. Hierdoor functioneert het RPN als een soort attention-mechanisme dat de zoekruimte voor objectdetectie beperkt. De integratie van regio-voorstellen binnen het netwerk is de grootste innovatie van Faster R-CNN en maakt het model aanzienlijk sneller dan zijn voorgangers.

RoI Pooling (of RoI Align in nieuwere implementaties) zorgt ervoor dat regio's van verschillende dimensies worden omgezet naar een vaste grootte, zodat ze verwerkt kunnen worden door de daaropvolgende fully connected lagen. Deze stap is essentieel voor de standaardisatie van input naar de classificatie- en regressieheads.

Tot slot bevat het model twee heads die parallel werken: het classificatiehead bepaalt de objectklasse binnen een regio en het regressiehead verfijnt de coördinaten van de **bounding box**. Deze dubbele aanpak zorgt voor nauwkeurige detectie en positionering van objecten. Ook wordt er gebruik gemaakt van NMS, die voorkomt dat meerdere **bounding boxes** rond hetzelfde object worden behouden.

De originele R-CNN had aanzienlijke beperkingen, waaronder de trage verwerking van duizenden regio's en een gefragmenteerd trainingsproces. Ook de afhankelijkheid van het traag werkende Selective Search-algoritme was een belangrijke bottleneck. Fast R-CNN verbeterde de snelheid door slechts één CNN-pass over de volledige afbeelding te doen en daarna RoI pooling toe te passen. Toch bleef het afhankelijk van externe regio-voorstellen, wat de algehele efficiëntie beperkte. Faster R-CNN loste deze beperking definitief op. Door het gebruik van een geïntegreerd RPN kon het model regio-voorstellen genereren als onderdeel van het netwerk zelf. Bovendien werd computationele efficiëntie verbeterd door convolutionele lagen te delen tussen het RPN en de detectiecomponent (Fast R-CNN). Dankzij deze integratie is het model end-to-end trainbaar, met aanzienlijke winst in zowel snelheid als nauwkeurigheid.

### SSD

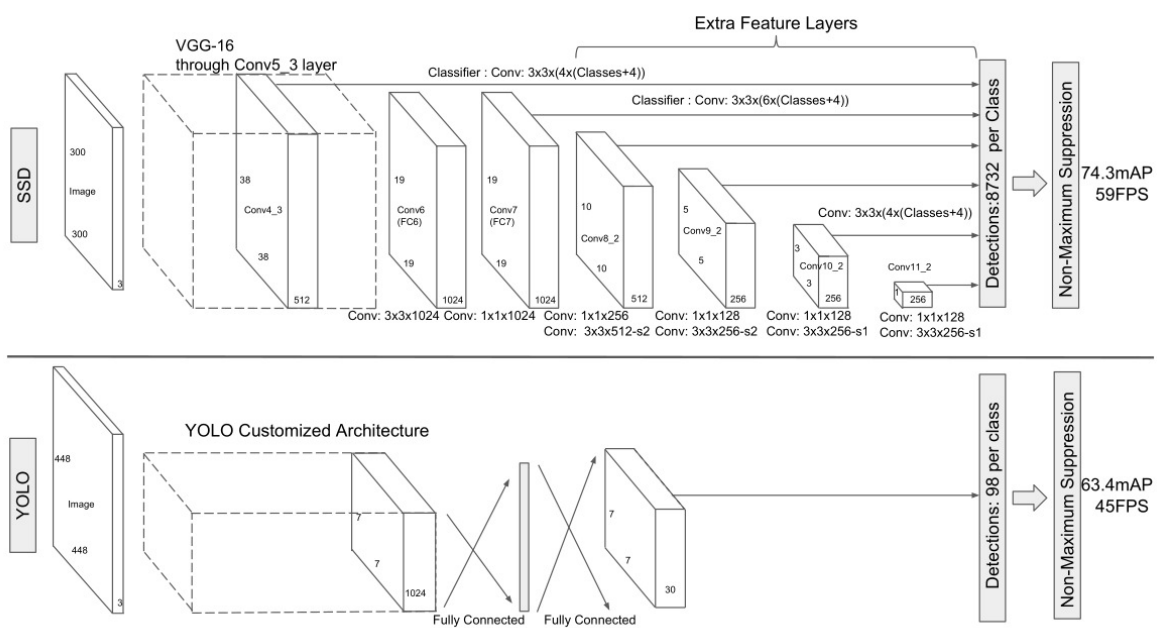
Een ander state-of-the-art model dat populair is binnen academische onderzoeken, is SSD. Deze architectuur werd voorgesteld in een artikel van Liu e.a. (2015). SSD is een zogenaamde single-stage detector die een goede balans biedt tussen nauwkeurigheid en snelheid. SSD detecteert objecten in één enkele pass door het netwerk, waarbij het gebruikmaakt van zogenaamde *multiboxes* – vooraf gedefinieerde rechthoeken die mogelijke objectlocaties voorstellen. Dit soort single-stage-architecturen zijn het gevolg van de vraag naar real-time beeldverwerking, zoals vereist in autonome voertuigen of slimme camera's. De geïntegreerde aanpak maakt SSD aanzienlijk sneller. Hoewel two-stage modellen over het algemeen preciezer zijn, vooral bij kleine objecten, biedt SSD een aantrekkelijk alternatief wanneer snelheid en eenvoud cruciaal zijn.

Het SSD-algoritme werkt met één enkel neurale netwerk dat tegelijk de objectcategorieën en de bijbehorende *bounding boxes* voorspelt. Het netwerk verdeelt het beeld in meerdere *default boxes* met verschillende schalen en beeldverhoudingen. Voor elke box voorspelt het de aanwezigheid van een object en past het de vorm van de box aan indien nodig. Belangrijk is dat SSD gebruikmaakt van meerdere lagen (feature maps) uit het netwerk: lagen met een hoge resolutie detecteren kleine objecten, terwijl diepere lagen grotere objecten beter kunnen herkennen. Deze multi-scale aanpak is de sleutel tot de flexibiliteit van SSD in het omgaan met objecten van verschillende groottes.

Een bijkomend voordeel van SSD is dat het geen aparte stap nodig heeft voor het genereren van objectvoorstellen, wat tijd bespaart. Tegelijkertijd voert het netwerk voor elke *bounding box* twee voorspellingen uit: één voor de objectklasse en één voor de precieze locatie. Deze voorspellingen worden gedaan via kleine convolutionele filters.

De SSD-architectuur bestaat uit een basisnetwerk voor feature extractie (zoals VGG16, ResNet of MobileNet), gevolgd door extra lagen die meerdere resoluties van het beeld analyseren. De oorspronkelijke implementatie gebruikte VGG16, maar varianten met lichtere of diepere netwerken zijn later ontwikkeld, afhankelijk van de toepassing. De keuze van het basisnetwerk bepaalt voor een groot deel de balans tussen nauwkeurigheid en gebruik van resources.

Omdat SSD veel overlappende voorspellingen kan genereren, is een extra stap nodig om dubbele detecties te vermijden. Dit gebeurt via NMS, een techniek die alleen de meest waarschijnlijke voorspelling voor elk object behoudt en overlappende alternatieven verwijdert. NMS is hierbij essentieel voor het produceren van duidelijke en bruikbare detectieresultaten.



**Figuur 2.5:** Vergelijking tussen de YOLO-architectuur en de SSD-architectuur. (Liu e.a., 2015)

SSD heeft een aantal belangrijke voordelen. Het is snel, eenvoudig en efficiënt in geheugengebruik en computationele resources. Dankzij het gebruik van meerdere feature maps kan het objecten van uiteenlopende groottes detecteren, en voor grote objecten is de nauwkeurigheid vaak uitstekend. Bovendien is de trainingsprocedure doorgaans sneller dan die van two-stage modellen.

Tegelijkertijd kent SSD ook beperkingen. De nauwkeurigheid bij het detecteren van kleine objecten blijft een uitdaging. Verder kunnen er tijdens training veel “achtergrondboxes” aanwezig zijn, wat het moeilijker maakt voor het model om onderscheid te maken tussen achtergrond en relevante objecten. Ondanks het gebruik van verschillende schalen, kan het model moeite hebben met objecten die aan-

zienlijk kleiner of groter zijn dan wat het tijdens training heeft gezien. Ook kunnen de voorspelde locaties minder precies zijn dan bij sommige andere methoden.

### 2.2.5. Specifieke toepassingen

Objectdetectie met deep learning is de afgelopen jaren breed toepasbaar gebleken in uiteenlopende domeinen, mede dankzij de ontwikkeling van de krachtige modellen hierboven besproken. Elk van deze modellen heeft specifieke sterktes die hen geschikt maken voor verschillende praktische toepassingen, afhankelijk van de vereisten op het vlak van snelheid, nauwkeurigheid en computationele resources.

**YOLO** is ontworpen met real-time toepassingen in gedachten en blinkt uit in snelheid zonder een al te grote toegeving in nauwkeurigheid. Hierdoor wordt het model veel ingezet in domeinen waar onmiddellijke beslissingen noodzakelijk zijn. (Dewan e.a., 2022) In autonome voertuigen wordt **YOLO** bijvoorbeeld gebruikt om verkeersborden, voetgangers en obstakels in real-time te detecteren, wat cruciaal is voor veilige navigatie. Ook in bewakingssystemen komt **YOLO** goed tot zijn recht: het detecteert onmiddellijk personen of objecten in videobeelden en maakt snelle interventies mogelijk. Daarnaast zijn er toepassingen in de gezondheidszorg, zoals het lokaliseren van tumoren in medische beelden, en in de landbouw, waar het model ingezet wordt voor het herkennen van gewasziekten of onkruid. Het onderzoek van C. Chen e.a. (2023) heeft aangetoond dat **YOLO** een praktisch nut biedt binnen onderwaterbeeldvorming. Daarnaast heeft het onderzoek van Bochkovskiy e.a. (2020) aangetoond dat de ontwikkeling van het model ervoor heeft gezorgd dat het één van de beste keuzes is geworden voor objectdetectie.

**SSD** biedt een compromis tussen snelheid en nauwkeurigheid en is daarmee eveneens geschikt voor real-time toepassingen, maar met iets betere detectieprestaties voor kleinere objecten dan **YOLO**. (Kumar & Srivastava, 2020) **SSD** wordt vaak ingezet in **AR**-toepassingen, waar het accuraat identificeren van objecten in de fysieke omgeving essentieel is om digitale elementen er realistisch op af te stemmen. Ook in autonome voertuigen wordt **SSD** gebruikt, bijvoorbeeld om bewegende objecten zoals voetgangers of andere auto's op tijd waar te nemen. In beveiligingstoepassingen helpt **SSD** bovendien bij het monitoren van camerabeelden om ongewenste activiteiten automatisch te signaleren. Het onderzoek van Ma e.a. (2020) heeft **SSD** bijvoorbeeld gebruikt om afval te detecteren. Ook binnen onderwaterobjectdetectie is **SSD** al enkele keren gebruikt, zoals in het onderzoek van Z. Jiang en Wang (2020).

Faster **R-CNN**, ten slotte, is een model dat zich onderscheidt door zijn hoge nauwkeurigheid, maar in ruil daarvoor minder geschikt is voor real-time toepassingen door de hogere computationele requirements. Het model wordt daarom veel gebruikt in situaties waar precisie belangrijker is dan snelheid. Zo wordt Faster **R-CNN** ingezet in lucht- en satellietbeelden voor het identificeren van gebouwen, voertuigen of vegetatie. In de medische beeldanalyse wordt het model toegepast voor het nauwkeurig lokaliseren van afwijkingen zoals tumoren of letsels, wat essentieel is voor een betrouwbare diagnose. Ook in industriële toepassingen, zoals kwaliteitscontrole op productielijnen, helpt Faster **R-CNN** bij het detecteren van defecten met een hoge mate van precisie. Ten slotte heeft Faster **R-CNN** zijn effectiviteit al bewezen in onderwaterobjectdetectie, zoals in het onderzoek van H. Wang en Xiao (2023) of Zeng e.a. (2021). Het onderzoek van Yulin e.a. (2020) heeft aangetoond dat het model ook op sonardata effectief is. Daarnaast is het model goed in het detecteren in kleine objecten (wat in sonardata vaak het geval is). Dit is gebleken uit het onderzoek van C. Chen e.a. (2017).

De keuze tussen deze modellen is sterk afhankelijk van de specifieke context waarin objectdetectie wordt toegepast. Terwijl **YOLO** en **SSD** uitblinken in snelheid voor dynamische omgevingen, biedt Faster **R-CNN** de precisie die vereist is in kritieke of specialistische domeinen zoals geneeskunde en geospatiale analyse.

## **2.3. Semi-supervised learning: principes en technieken**

### **2.3.1. Definitie en waarde binnen objectdetectie**

Binnen het domein van machine learning bestaan er verschillende technieken om een model te trainen. Meestal wordt er gesproken van twee grote stromingen: supervised learning en unsupervised learning. Bij supervised learning wordt er gebruik gemaakt van een dataset en een uitkomst (hetgeen het model uiteindelijk moet kunnen voorspellen). Dit kan een label zijn of een bepaalde numerieke waarde. Belangrijk is dat zowel de input als de gewenste output gegeven zijn. Het model leert dus het verband tussen de twee. Bij unsupervised learning zijn er geen verwachte outputs. De volledige dataset wordt door het model gebruikt om patronen in te herkennen. Unsupervised learning wordt daarom ook meestal gebruikt om verkennende data-analyse uit te voeren. Echter hebben beide methoden enkele nadelen. Bij supervised learning is het traag en duur om alle data op een correcte manier te labelen. Unsupervised learning heeft dit probleem niet, maar heeft een beperkt aantal toepassingen en is minder accuraat. Een alternatief is **SSL**, wat een compromis tussen zowel supervised als unsupervised learning is. (C A Padmanabha Reddy e.a., 2018)

**SSL** is een subdomein van machine learning waar gebruik gemaakt wordt van zowel gelabelde als ongelabelde data om modellen te trainen. Dit is bijzonder nuttig in situaties waarin het labelen van gegevens duur of tijdrovend is, zoals bij sonar-data het geval is. **SSL** bevindt zich tussen supervised learning (waar alle trainingsdata gelabeld zijn) en unsupervised learning (waar geen labels beschikbaar zijn). Door gebruik te maken van een kleine hoeveelheid gelabelde gegevens in combinatie met een grote hoeveelheid ongelabelde gegevens, kan een model beter generaliseren waardoor de prestaties verbeteren met minder menselijke annotatie-inspanning. (Hady & Schwenker, 2013)

Zoals eerder vermeld, is **SSL** een subdomein van machine learning, net zoals supervised en unsupervised learning. Het is dus niet één model of één architectuur, maar een waaier van verschillende algoritmen die soms op hele andere manieren werken. Wel hebben ze allemaal gemeen dat ze zowel gelabelde als ongelabelde data gebruiken.

Één van de belangrijkste technieken binnen **SSL** is *consistency regularization* (cf. infra), waarbij een model wordt aangemoedigd om consistente voorspellingen te maken voor kleine verstoringen van dezelfde ongelabelde input. (Fan e.a., 2022) Een ander veelgebruikt principe is *pseudo-labeling* (cf. infra), waarbij het model zelf voorspellingen genereert voor ongelabelde gegevens en deze gebruikt als extra trainingsdata. (D.-H. Lee, 2013)

Daarnaast is er graph-based **SSL**. Dit is een techniek die gebruik maakt van grafen (netwerken) om de relaties tussen gelabelde en ongelabelde data te modelleren en labelinformatie effectiever te verspreiden. In plaats van uitsluitend te vertrouwen op individuele gegevenspunten, gebruiken deze methoden de structuur van de dataset om aannames te maken over onbekende labels. Dit is vooral nuttig in situaties waarin de onderliggende data een natuurlijke connectiviteit vertoont, zoals sociale netwerken, biologische netwerken en tekstanalyses. (Song e.a., 2021)

Graph-based **SSL**-modellen stellen de dataset voor als een graaf  $G = (V, E)$  waarbij  $V$  de knopen (datapunten) zijn (zowel gelabelde als ongelabelde gegevens) en  $E$  de gewogen randen (connecties) tussen knopen zijn, die de relatie of gelijkens tussen de datapunten aangeven. Het basisidee is dat naburige knopen waarschijnlijk tot dezelfde klasse behoren, een principe dat bekend staat als *label propagation*. Labels van bekende knopen (gelabelde data) worden hierbij iteratief verspreid naar naburige knopen op basis van de sterkte van de verbindingen. (Zhu, 2005)

**SSL** wordt toegepast in diverse domeinen, zoals beeld- en spraakherkenning, bio-medische analyse en autonome systemen. Recente ontwikkelingen in deep learning

hebben geleid tot geavanceerde **SSL**-methoden, zoals FixMatch en MixMatch (cf. infra), die de prestaties aanzienlijk verbeteren door sterke data-augmentatie, efficiënter gebruik van ongelabelde data en de combinatie van andere **SSL**-technieken. Onderzoek heeft aangetoond dat **SSL** met slechts 10% gelabelde data al bijna dezelfde prestaties kan bereiken als volledig gelabelde modellen. (Lucas e.a., 2022)

### **2.3.2. Veelgebruikte SSL-methoden**

**Semi-Supervised Learning (SSL)** omvat verschillende methoden die gebruik maken van zowel gelabelde als ongelabelde data om de prestaties van machine learning-modellen te verbeteren. Veelgebruikte **SSL**-methoden compenseren de beperkte beschikbaarheid van gelabelde data door patronen en structuren in ongelabelde data te benutten. **SSL** wordt voor alle soorten doeleinden gebruikt: niet alleen in predictieve modellen, waarbij er vaak gebruik gemaakt wordt van pseudo-labeling en consistency regularization, wordt **SSL** ingezet. Ook bij generatieve modellen wordt **SSL** enorm veel gebruikt. Dit gaat dan bijvoorbeeld om **Variational Autoencoders (VAEs)** en **Generative Adversarial Networks (GANs)**. Deze worden hierbij ingezet om aanvullende trainingsgegevens te genereren. Door al deze technieken te combineren, kunnen **SSL**-methoden significante verbeteringen bieden voor taken zoals beeldherkenning, spraakverwerking en **Natural Language Processing (NLP)**. (van Engelen & Hoos, 2019) Hieronder worden enkele van de meest prominente technieken binnen **SSL** besproken.

#### **Pseudo-labeling**

Pseudo-labeling is een belangrijke techniek binnen **SSL** waarbij een model, getraind op een beperkte hoeveelheid gelabelde data, wordt ingezet om voorspellingen te doen op ongelabelde data. Deze voorspellingen, aangeduid als “pseudo-labels”, worden vervolgens behandeld als echte labels, waardoor het model verder kan worden verfijnd met een uitgebreidere dataset. Dit proces wordt iteratief herhaald, zodat het model geleidelijk aan zijn prestaties verbetert door zowel de gelabelde als de pseudo-gelabelde data te gebruiken. (D.-H. Lee, 2013)

Het succes van pseudo-labeling is natuurlijk afhankelijk van de nauwkeurigheid van de gegenereerde pseudo-labels. Om de kwaliteit te waarborgen, wordt vaak een drempelwaarde ingesteld voor de voorspellingszekerheid: alleen voorspellingen die boven deze drempel uitkomen, worden als pseudo-labels geaccepteerd. Dit helpt het model om zich te concentreren op voorbeelden waarbij het relatief zeker is van de voorspelling, waardoor het risico op het leren van verkeerde informatie wordt verminderd. (Kage e.a., 2024)

Een belangrijk voordeel van pseudo-labeling is dat het effectief gebruikmaakt van grote hoeveelheden ongelabelde data, wat vooral nuttig is in domeinen zoals sonar-beeldvorming, waar het verkrijgen van gelabelde data duur en tijdrovend is. Verder

zijn toepassingen van pseudo-labeling te vinden in verschillende gebieden, waaronder beeldherkenning, spraakverwerking en NLP. (Min e.a., 2022) Recent onderzoek van Ferreira e.a. (2023) heeft aangetoond dat pseudo-labeling een zeer goede performantie kan neerzetten, terwijl het de behoefte aan uitgebreide gelabelde datasets vermindert.

Desondanks kent pseudo-labeling ook uitdagingen. Als het model in een vroeg stadium onnauwkeurige pseudo-labels genereert, kan dit leiden tot het versterken van fouten, een fenomeen bekend als *confirmation bias*. Om dit te voorkomen, worden technieken zoals *curriculum learning* toegepast, waarbij het model eerst wordt getraind met de meest zekere pseudo-labels en geleidelijk aan minder zekere voorbeelden toevoegt naarmate de training vordert. (Cascante-Bonilla e.a., 2020)

### Consistency Regularization

Een ander fundamenteel concept binnen SSL is consistency regularization. Deze techniek wordt in verschillende algoritmen gebruikt om de betrouwbaarheid van het model te verbeteren. Het doet dit door het model te dwingen consistente voorspellingen te maken voor kleine variaties van dezelfde input. Het idee is gebaseerd op de veronderstelling dat een model robuust moet zijn tegen kleine verstoringen in de input, vooral wanneer de input geen gelabelde gegevens bevat.

Eerst wordt een ongelabeld voorbeeld  $x_u$  aangepast met kleine verstoringen. Dit kan gaan om data-augmentatie (bv. willekeurige rotaties, verscherping, kleuraanpassingen, ...), het toevoegen van ruis (bv. Gaussian noise), ...Dit resulteert in twee versies van dezelfde input: het origineel  $x_u$  en de verstoorte versie  $x'_u$ . Daarna voorspelt het model de kansverdeling van de klassen voor zowel de originele als de verstoorte input.

Een consistente voorspelling betekent dat beide kansverdelingen "dicht" bij elkaar moeten liggen. Om dit te garanderen wordt een gespecialiseerde *loss-functie* gebruikt, zoals de *Kullback-Leibler (KL)-divergentie* (Hall, 1987) of de *Mean Squared Error (MSE)*. Het model wordt getraind om deze *loss* te minimaliseren, zodat het stabiele en robuuste voorspellingen leert maken, zelfs bij verstoringen.

Consistency regularization is enorm effectief (en wordt daarom ook veel gebruikt) omdat het ervoor zorgt dat het model gebruik maakt van de onderliggende structuur van ongelabelde data. Hierdoor verbetert de generalisatie, omdat het model minder gevoelig wordt voor kleine ruis en variaties. Daarnaast verhoogt de sample-efficiëntie, waardoor minder gelabelde data nodig is voor goede prestaties. (Fan e.a., 2022)

### FixMatch

Beide voorgaande technieken – zowel pseudo-labeling als consistency regularization – zijn slechts concepten binnen het domein van SSL. Dit wil daarom niet zeggen dat ze niet gebruikt worden in productie, maar vaak zijn ze slechts steunpilaren voor een grotere, complexere en effectievere architecturen. Hieronder wordt FixMatch – een eerste voorbeeld die relevant is voor dit onderzoek – besproken.

FixMatch combineert pseudo-labeling met consistency regularization. Het werd geïntroduceerd door Sohn e.a. (2020) en heeft bewezen indrukwekkende prestaties te leveren op benchmarks zoals CIFAR-10, waarbij met slechts 250 gelabelde voorbeelden een nauwkeurigheid van 94,93% werd bereikt. De kernideeën achter FixMatch zijn gebaseerd op het principe dat een model consistente voorspellingen zou moeten maken, zelfs wanneer invoerdata wordt gewijzigd met verschillende vormen van data-augmentatie. FixMatch combineert twee belangrijke technieken binnen SSL: pseudo-labeling en sterke versus zwakke augmentatie.

Voor een gegeven ongelabeld voorbeeld  $x_u$  wordt eerst een zwak geaugmenteerde versie  $x_u^w$  gegenereerd met eenvoudige transformaties zoals willekeurige spiegelingen en verschuivingen. Het model maakt een voorspelling  $p(y|x_u^w)$ , wat een kansverdeling over de mogelijke klassen oplevert. Als de hoogste voorspelde waarschijnlijkheid boven een drempel  $\tau$  ligt wordt dit label als pseudo-label gebruikt.

$$\hat{y}_u = \arg \max p(y|x_u^w) \text{ indien } \max p(y|x_u^w) > \tau$$

Dit voorkomt dat onzekere voorspellingen als pseudo-labels worden gebruikt, waardoor ruis in de trainingsdata wordt verminderd.

Vervolgens wordt een sterk geaugmenteerde versie  $x_u^s$  van dezelfde input  $x_u$  gemaakt met agressieve augmentaties, zoals kleurvervormingen en Cutout (willekeurige afdekking van delen van de afbeelding). (DeVries & Taylor, 2017) Het model wordt getraind om dezelfde voorspelling  $\hat{y}_u$  te maken op de sterk geaugmenteerde input  $x_u^s$ , waardoor het robuust wordt tegen verschillende variaties in de data.

Om FixMatch te trainen is er echter een gespecialiseerde loss-functie nodig, die een combinatie is voor zowel de gelabelde als de pseudo-gelabelde data:

$$L = L_s + \lambda_u L_u$$

waarbij  $L_s$  de gewone loss-functie is voor de gelabelde data,  $L_u$  de consistency loss is voor pseudo-gelabelde data, gedefinieerd als de gemiddelde cross-entropy tussen het pseudo-label en de voorspelling op  $x_u^s$ , en  $\lambda_u$  een hyperparameter is die de balans tussen gelabelde en ongelabelde data regelt.

De hoge performantie van FixMatch heeft verschillende redenen: in tegenstelling tot traditionele pseudo-labeling, waarbij alle voorspellingen worden gebruikt, accepteert FixMatch alleen pseudo-labels als de modelvoorspelling een zekerheid boven een vooraf ingestelde drempel  $\tau$  heeft. Dit voorkomt het versterken van foute pseudo-labels en verbetert de robuustheid van het model. Ook wordt het model gedwongen om dezelfde voorspelling te maken voor een input, ongeacht of deze zwak of sterk geaugmenteerd is. Dit helpt het model om stabiele, generaliseerbare representaties te leren. Daarnaast voorkomt het toepassen van een tweefasige-augmentatiestrategie dat het model te afhankelijk wordt van specifieke kenmerken in de data. Dit helpt overfitting te verminderen.

### MixMatch

MixMatch integreert ook verschillende SSL-technieken, waaronder data-augmentatie, pseudo-labeling en mixup. Het algoritme is een verbetering op de FixMatch-architectuur en werd geïntroduceerd door Berthelot e.a. (2019). In benchmarks heeft het hele goede prestaties geleverd op datasets zoals CIFAR-10 en STL-10.

Voor elk ongelabeld voorbeeld wordt het model meerdere keren toegepast om voorspellingen te genereren. Het gemiddelde van de voorspellingen wordt gebruikt om ruis te verminderen en betrouwbaardere pseudo-labels te verkrijgen. Dit helpt het model om robuustere pseudo-labels te creëren dan traditionele pseudo-labeling, waar slechts één voorspelling per voorbeeld wordt gebruikt.

Om ervoor te zorgen dat de pseudo-labels niet te onzeker zijn, wordt een *softmax-temperature-scaling* toegepast om de kansverdeling scherper te maken. Dit betekent dat het model meer vertrouwen krijgt in de meest waarschijnlijke klassen, waardoor het pseudo-label effectiever wordt.

$$q(y|x) = \frac{p(y|x)^{\frac{1}{T}}}{\sum_{y'} p(y'|x)^{\frac{1}{T}}}$$

waarbij  $T$  de temperatuurparameter is die de scherpte van de distributie regelt (kleinere waarden leiden tot scherpere distributies).

MixMatch maakt ook gebruik van *mixup*, een data-augmentatietechniek waarbij synthetische trainingssamples worden gemaakt van paren bestaande trainingssamples en hun labels  $((x_1, y_1)$  en  $(x_2, y_2))$ . Merk op dat zowel de gelabelde als de pseudo-gelabelde samples gebruikt worden.

$$\tilde{x} = \lambda x_1 + (1 - \lambda)x_2$$

$$\tilde{y} = \lambda y_1 + (1 - \lambda)y_2$$

Hierbij is  $\lambda$  een gewichtsparemeter, die willekeurig gekozen is uit een Beta-verdeling. Deze techniek zorgt voor soepelere beslissingsgrenzen en vermindert overfitting door het model te dwingen om robuuster te generaliseren. (Zhang e.a., 2017)

Omdat dezelfde input meerdere keren wordt geaugmenteerd en verwerkt, wordt het model getraind om consistente voorspellingen te maken voor deze variaties. Dit dwingt het model om stabiele representaties te leren, zelfs wanneer kleine veranderingen worden aangebracht in de inputdata. Net zoals FixMatch minimaliseert MixMatch een gecombineerde **loss-functie** die zowel de gelabelde als de pseudo-gelabelde voorbeelden gebruikt (cf. supra). (Berthelot e.a., 2019)

## **2.4. Self-supervised learning: principes en technieken**

### **2.4.1. Definitie en verschil met SSL**

Anders dan bij **SSL** is er bij **Self-SL** helemaal geen nood aan labels. Deze creëert het algoritme namelijk zelf tijdens een pre-training fase. **Self-SL** behoort echter niet helemaal tot het domein van unsupervised learning, hoewel het gebruik maakt van verschillende groeperings- en clusteringmethoden. Het uiteindelijke model maakt namelijk gebruik van – door de pre-training – gelabelde data. Deze techniek zorgt ervoor dat er veel complexere modellen getraind kunnen worden zonder een gigantische hoeveelheid aan data. Enkele nadelen zijn wel dat de techniek een grote hoeveelheid computerkracht nodig heeft en een lagere accuratie heeft dan supervised learning. (Gui e.a., 2024)

Het fundamentele verschil tussen **Self-SL** en **SSL** ligt in de manier waarop ze omgaan met labels. In **SSL** zijn er externe, door mensen geannoteerde labels aanwezig, zij het in beperkte mate. **Self-SL** genereert daarentegen volledig zijn eigen labels zonder externe annotaties. Hierdoor wordt **Self-SL** soms beschouwd als een vorm van unsupervised learning, maar met expliciet gedefinieerde pretext-taken om de onderliggende structuren in data beter te benutten.

Ondanks deze nadelen blijken verschillende populaire technieken succesvol binnen het domein van computer vision.

### **2.4.2. Werking**

**Self-SL** probeert dus structuur, patronen of representaties te leren uit ruwe data door artificiële taken te formuleren die als “surrogaat”-targets dienen. De fundamentele werking van self-supervised learning steunt op het concept van pretext-taken: eenvoudige, automatisch afleidbare taken die een model dwingen om zinvolle representaties te leren. Voorbeelden van zulke taken zijn het voorspellen van het ontbrekende deel van een afbeelding (*inpainting*), het reconstrueren van een

permutatie van patches (*jigsaw puzzles*), of het bepalen van de relatieve positie tussen twee afbeeldingsfragmenten. Hoewel het model op deze taken niet direct getraind wordt om het uiteindelijke doel (zoals classificatie of detectie) te bereiken, leert het wél onderliggende semantische structuren en contexten herkennen – wat waardevol blijkt bij finetuning voor downstream-taken.

In recente jaren is er een verschuiving ontstaan richting contrastieve en predictieve methodes binnen *Self-SL*. Contrastive learning probeert gelijkaardige samples dicht bij elkaar te brengen in de embedding-ruimte en verschillende samples uit elkaar te duwen. (Oord e.a., 2018) Bekende frameworks zoals *SimCLR* en *Momentum Contrast (MoCo)* gebruiken deze techniek. *BYOL*, daarentegen, doet afstand van expliciete negatieve paren en vertrouwt op twee netwerken – een online netwerk en een target netwerk – die via momentum-updates met elkaar samenwerken (cf. infra). Het model leert door representaties van verschillende augmentaties van hetzelfde beeld op elkaar af te stemmen. Hierdoor leert het betekenisvolle representaties zonder dat negatieve voorbeelden nodig zijn.

Het einddoel van *Self-SL* is het trainen van een krachtige feature-extractor – of backbone – die generaliseerbare en contextuele representaties levert. Deze backbone kan daarna gebruikt worden in taken zoals objectdetectie, segmentatie of classificatie. In veel toepassingen, zoals bij het pretrainen van een convolutioneel neuraal netwerk voor gebruik in *Faster R-CNN*, is *Self-SL* een robuust alternatief voor pre-training op volledig gelabelde datasets. Daarmee maakt *Self-SL* het mogelijk om beter gebruik te maken van grote hoeveelheden ruwe data – een cruciale troef in domeinen waar gelabelde data schaars is.

### 2.4.3. Veelgebruikte *Self-SL* methoden

In de context van vooruitgang in *SSL* zijn er verschillende praktische modellen ontwikkeld die de eerder besproken theoretische principes – zoals contrastive learning en representatieleren zonder gelabelde data – succesvol toepassen op grootschalige visuele data. Deze modellen onderscheiden zich niet alleen door hun architecturale keuzes, maar ook door hoe ze omgaan met negatieve samples, augmentatiestrategieën en het gebruik van momentum-updates. In deze sectie worden enkele invloedrijke en veelgebruikte *SSL*-methoden besproken in realistische settings, met een focus op *Simple Framework for Contrastive Learning of Visual Representations (SimCLR)*, *Momentum Contrast (MoCo)* en *Bootstrap Your Own Latent (BYOL)*. Elk van deze modellen illustreert op unieke wijze hoe self-supervised representaties kunnen worden geleerd met minimale supervisie, en vormt daarmee een belangrijke bouwsteen in moderne computer vision toepassingen.

### SimCLR

Een bekende Self-SL-techniek is SimCLR. Deze afkorting staat voor *Simple Framework for Contrastive Learning of Visual Representations*. Met andere woorden maakt deze techniek dus gebruik van *contrastive learning* om visuele representaties te leren zonder de noodzaak van gelabelde data. Het werd ontwikkeld door onderzoekers van Google Brain en gepresenteerd in een paper van T. Chen e.a. (2020). Het doel van SimCLR is om een neurale netwerk zodanig te trainen dat het visuele representaties van afbeeldingen leert door contrastieve relaties te benutten.

Om een model contrastieve representaties te laten leren, worden er van elke invoerafbeelding eerst twee willekeurige transformaties gemaakt. Deze transformaties kunnen bestaan uit verschillende dingen, waaronder:

- Willekeurig bijsnijden en schalen
- Kleurveranderingen zoals de helderheid en contrast aanpassen
- Toepassen van blurring
- Rotatie of horizontale spiegeling

Deze transformaties zorgen ervoor dat het model leert om dezelfde afbeelding te herkennen, ongeacht variaties in uiterlijk. Na de augmentaties worden de twee versies van de afbeelding door een encoder gestuurd, meestal een CNN (zoals een ResNet). Dit netwerk zet de invoerafbeeldingen om in zogenaamde *feature vectors* die de kernkenmerken van de afbeelding representeren.

De gegenereerde feature vectors worden vervolgens door een *projection head* gestuurd. Dit is een klein neurale netwerk dat de feature vector transformeert naar een ruimte waarin de contrastieve vergelijking plaatsvindt (latente ruimte). Dit projection head bestaat meestal uit een paar fully connected of dense-lagen en wordt na training weggegooid, omdat alleen de encoder nodig is voor downstream taken (zoals beeldclassificatie, objectdetectie en semantische segmentatie). (Gupta e.a., 2022)

Het doel van SimCLR is om representaties van verschillende augmentaties van dezelfde afbeelding dicht bij elkaar te brengen en representaties van verschillende afbeeldingen verder uit elkaar te duwen. Dit gebeurt met behulp van de *Normalized Temperature-scaled Cross Entropy (NT-Xent)* loss. De *loss-functie* wordt berekend zodat positieve paren (twee augmentaties van dezelfde afbeelding) een hoge gelijkheid hebben en negatieve paren (verschillende afbeeldingen in de *batch*) een lage gelijkheid. De cosinusgelijkheid wordt vaak gebruikt om de afstand tussen de verschillende vectoren te meten. Daarnaast beïnvloedt de temperatuurparameter  $\tau$  in de *NT-Xent* loss hoe streng de loss reageert op verschillen in gelijkheid tussen paren.

### MoCo

Een andere veelgebruikte Self-SL-techniek is MoCo. Deze methode maakt gebruik van contrastief leren om visuele representaties te leren zonder de noodzaak van gelabelde data. Het werd geïntroduceerd in een paper van He e.a. (2019) en heeft sindsdien aanzienlijke aandacht gekregen binnen het domein van de computer vision. MoCo introduceert verschillende innovatieve technieken zoals het *momentum update*-mechanisme en een dynamisch woordenboek om contrastief leren te verbeteren. Ook wordt er een gespecialiseerde *loss-functie* gebruikt die de basis vormt van het leerproces. Dankzij deze strategieën kan MoCo superieure representaties leren zonder gelabelde data, wat het een krachtige Self-SL-techniek maakt.

Het MoCo-framework bestaat uit meerdere essentiële componenten die samenwerken om effectieve visuele representaties te leren. De *query encoder* ( $f_q$ ) is verantwoordelijk voor het omzetten van een *query sample* (zoals een geaugmenteerde versie van een afbeelding) in een *feature vector*. De parameters van deze encoder worden bijgewerkt via standaard backpropagation, waarbij de optimalisatie wordt gestuurd door een contrastieve *loss-functie*. (Sowe & Bah, 2025)

Daarnaast bevat MoCo een *momentum key encoder* ( $f_k$ ) die wordt gebruikt om *keys* (bijvoorbeeld geaugmenteerde weergaven uit eerdere *mini-batches*) om te zetten in *feature vectors*. Een cruciale innovatie van MoCo is het *momentum update*-mechanisme. In plaats van direct via backpropagation te worden bijgewerkt, worden de parameters van de *key encoder* ( $\theta_k$ ) aangepast als een voortschrijdend gemiddelde van de *query encoder* ( $\theta_q$ ). Dit gebeurt volgens de volgende formule:

$$\theta_k \rightarrow m\theta_k + (1 - m)\theta_q$$

Hierbij is  $m$  de *momentumcoëfficiënt*, die doorgaans dicht bij 1 ligt (bv. 0,999). Dit zorgt voor een geleidelijke en stabiele update van de key encoder, waardoor de representaties van de keys na verloop van tijd minder variëren. Een stabielere woordenboek van negatieve samples is cruciaal voor contrastief leren, omdat het helpt om robuuste en onderscheidende kenmerken te leren.

Om een grote en gevarieerde set van negatieve samples te behouden, maakt MoCo gebruik van een dynamisch woordenboek dat wordt beheerd via een FIFO (First-In, First-Out) wachtrij. Dit mechanisme zorgt ervoor dat de grootte van het woordenboek niet wordt beperkt door de mini-batchgrootte, zoals bij traditionele contrastieve leermethoden het geval is.

- Wanneer een nieuwe mini-batch wordt verwerkt, worden de bijbehorende representaties aan het einde van de wachtrij toegevoegd (enqueue).

- Tegelijkertijd worden de oudste representaties uit de wachtrij verwijderd (de-queue).

Dit ontkoppelt het aantal negatieve samples van de batchgrootte, waardoor MoCo kan werken met veel grotere negatieve sets zonder dat dit een grote hoeveelheid GPU-geheugen vereist. Een groter en gevarieerder aantal negatieve samples helpt het model om betere representaties te leren, omdat de contrastieve taak uitdagender wordt. Dit dwingt het model om robuustere en meer onderscheidende kenmerken te ontwikkelen.

MoCo maakt gebruik van de InfoNCE loss, een contrastieve verliesfunctie die het model leert om:

1. De gelijkheid tussen een query en zijn bijbehorende positieve key te maximaliseren (dit zijn verschillende augmentaties van dezelfde afbeelding).
2. De gelijkheid tussen de query en negatieve keys te minimaliseren (deze representeren andere afbeeldingen in de wachtrij).

De wiskundige formulering van de InfoNCE loss is als volgt:

$$L_q = -\log \frac{\exp(q \cdot k^* / \tau)}{\sum_{i=0}^K \exp(q \cdot k_i / \tau)}$$

Hierbij geldt:

- $q$  = de gecodeerde query
- $k^*$  = de gecodeerde positieve key
- $k_i$  = alle keys in het woordenboek (inclusief  $k^*$  en  $K$  negatieve keys)
- $\tau$  = de temperatuurparameter, die de scherpte van de loss controleert

De InfoNCE loss vormt de kern van het leerproces van MoCo en stelt het model in staat om een embedding space te creëren waarin augmentaties van dezelfde afbeelding dicht bij elkaar liggen en representaties van verschillende afbeeldingen ver uit elkaar worden geplaatst. Dit proces, bekend als *instance discrimination*, helpt het model om semantisch betekenisvolle kenmerken te leren zonder gelabelde data.

### BYOL

Naast de contrastieve methoden die veelal gebruikt worden binnen het domein van Self-SL is er sinds enkele jaren ook een ander alternatief. BYOL is ontwikkeld bij Google DeepMind en werd geïntroduceerd in een paper van Grill e.a. (2020). BYOL is gebaseerd op het idee dat een model kan leren door zijn eigen representaties te

gebruiken als leersignaal. Dit gebeurt door twee verschillende, geaugmenteerde versies van hetzelfde inputbeeld te genereren, en het model leert vervolgens om de representatie van de ene versie te voorspellen op basis van de andere. In tegenstelling tot contrastieve leermethoden – die gebruikmaken van zowel positieve als negatieve paren – werkt **BYOL** volledig zonder negatieve voorbeelden. Terwijl contrastieve methoden proberen representaties van verschillende beelden uit elkaar te duwen, focust **BYOL** zich enkel op het dichter bij elkaar brengen van representaties van dezelfde onderliggende input. Deze aanpak vermindert de kans op instabiliteit en maakt het model robuuster voor variaties in data-augmentatie.

Het leerproces in **BYOL** is gebouwd op twee samenwerkende netwerken: een *online network* en een *target network*. Het online netwerk probeert de representatie te voorspellen die door het target netwerk is gegenereerd. Het target netwerk zelf wordt niet direct getraind via **backpropagation**, maar geüpdatet via een **Exponential Moving Average (EMA)** van de parameters van het online netwerk. Deze samenwerking tussen de twee netwerken zorgt voor een geleidelijke en stabiele leeromgeving.

Zoals hierboven vermeld, bestaat de **BYOL**-architectuur dus twee identieke netwerken qua structuur: het online netwerk en het target netwerk. Het online netwerk bevat drie componenten:

1. **Encoder ( $f_\theta$ ):** extraheert een representatie uit een geaugmenteerd inputbeeld.
2. **Projector ( $g_\theta$ ):** projecteert deze representatie naar een lagere-dimensionale vectorruimte.
3. **Predictor ( $q_\theta$ ):** probeert de representatie van het target netwerk te voorspellen.

Het target netwerk bevat alleen de encoder en projector, zonder predictor. Het wordt bijgehouden als een langzame, voortschrijdende kopie van het online netwerk, waardoor het stabielere leerdoelen biedt. Tijdens training worden twee verschillende augmentaties van eenzelfde afbeelding verwerkt: één door het online netwerk en de andere door het target netwerk. Het doel is dat de output van de predictor van het online netwerk overeenkomt met de projectie van het target netwerk.

De **loss-functie** voor **BYOL** is gebaseerd op de **MSE** tussen L2-genormaliseerde vectoren: de voorspelling van het online netwerk en de projectie van het target netwerk. (van Laarhoven, 2017) Voor een enkel augmentatiepaar  $(v, v')$  wordt de loss als volgt berekend:

$$L_{\theta,\xi} = \left\| \text{normalize}(q_{\theta}(z_{\theta})) - \text{normalize}(z'_{\xi}) \right\|_2^2 = 2 - 2 \cdot \langle \text{normalize}(q_{\theta}(z_{\theta})), \text{normalize}(z'_{\xi}) \rangle$$

Data-augmentatie speelt een centrale rol in **BYOL**. Door willekeurige transformaties toe te passen (zoals crops, kleurvervormingen, flips, ...), ontstaan twee verschillende maar inhoudelijk gelijke views van hetzelfde beeld. Deze variatie dwingt het model om representaties te leren die geen rekening houden met oppervlakkige veranderingen en gericht zijn op de semantische kern van het beeld. Interessant is dat **BYOL** minder gevoelig is voor de specifieke keuze van augmentaties dan contrastieve methoden zoals **SimCLR**. Het model blijft effectief presteren, zelfs wanneer bepaalde augmentaties worden weggelaten, wat wijst op een grotere robuustheid.

## 2.5. Vergelijking van relevante methoden

In dit onderzoek zijn verschillende leermethoden onderzocht binnen drie bredere categorieën: supervised learning, **SSL**, en **Self-SL**. Binnen elke categorie zijn meerdere representatieve modellen geëvalueerd op basis van hun geschiktheid voor objectdetectie in sonardata, met aandacht voor aspecten zoals precisie, robuustheid, data-efficiëntie, en computationele vereisten.

### 2.5.1. Gesuperviseerde architecturen

Binnen het domein van supervised objectdetectie zijn **YOLO**, **Faster R-CNN** en **SSD** drie toonaangevende architecturen met uiteenlopende sterktes en toepassingsgebieden. Alle drie zijn ontworpen om objectdetectie mogelijk te maken in volledig gelabelde datasets, maar ze verschillen aanzienlijk in hun aanpak van snelheid, nauwkeurigheid en architecturale complexiteit.

**YOLO** is een single-stage detector die objectdetectie als een regressietaak benadert, waarbij alle voorspellingen in één stap worden gedaan. Dit resulteert in zeer snelle inferentie, waardoor **YOLO** bijzonder geschikt is voor real-time toepassingen. De keerzijde is echter dat deze snelheid gepaard gaat met een lagere precisie bij het lokaliseren van kleine of dicht op elkaar liggende objecten – een mogelijke beperking bij complexere sonarbeelden.

**Faster R-CNN** daarentegen maakt gebruik van een tweefasige detectiestructuur. Eerst genereert het **RPN** potentiële objectlocaties, waarna een tweede netwerk verantwoordelijk is voor classificatie en verfijning van de **bounding boxes**. Hoewel deze aanpak leidt tot een hogere inferentietijd, is de nauwkeurigheid doorgaans beter, vooral in complexe en ruisgevoelige beelden. Dit maakt **Faster R-CNN** conceptueel aantrekkelijker voor toepassingen waarin precisie belangrijker is dan snelheid, zoals bij de analyse van onderwaterbeelden via sonar.

**SSD** neemt eveneens een single-shot benadering, maar introduceert verbeteringen zoals meerdere schalen en ankerboxen, wat de nauwkeurigheid enigszins verhoogt ten opzichte van vroege **YOLO**-versies. **SSD** biedt een evenwicht tussen snelheid en precisie, maar is nog steeds gevoeliger voor objecten met onregelmatige vormen of weinig contrast – karakteristiek voor sonaromgevingen.

Eigenschap	YOLO	Faster R-CNN	SSD
Architectuur	Single-shot	Two-stage	Single-shot
Nauwkeurigheid	Gematigd	Hoog	Gemiddeld
Snelheid	Zeer snel	Traag	Snel
Robuustheid bij ruis	Beperkt	Hoog	Matig

**Tabel 2.1:** Tabel met een simpele vergelijking tussen de onderzochte supervised modellen binnen dit onderzoek.

### 2.5.2. Semi-supervised architecturen

FixMatch en MixMatch vormen binnen **SSL** twee representatieve methoden die elk op een eigen manier trachten om het leerproces te verbeteren met behulp van een beperkte hoeveelheid gelabelde data en een grotere hoeveelheid ongesuperviseerde samples. Beide modellen zijn gebaseerd op het idee van consistency regularization – het principe dat het model consistente voorspellingen moet doen voor verschillende transformaties van dezelfde input – maar implementeren dit op fundamenteel verschillende manieren.

MixMatch combineert meerdere technieken in één framework: het genereert soft pseudo-labels via averaging, past entropy minimization toe om de outputdistributie te verscherpen, en gebruikt een MixUp-strategie om input en labelparen te interpoleren. Deze combinatie kan krachtige resultaten opleveren, vooral bij complexe data, maar introduceert ook aanzienlijke modelcomplexiteit en vereist precieze tuning van hyperparameters om optimale prestaties te behalen. Dit kan een uitdaging vormen in domeinspecifieke contexten zoals sonar, waar de optimale instellingen niet altijd intuïtief zijn.

FixMatch daarentegen stelt eenvoud centraal: het gebruikt een combinatie van pseudo-labeling op zwak geaugmenteerde voorbeelden, en eist vervolgens consistentie op sterk geaugmenteerde versies van dezelfde input – mits het model voldoende vertrouwen heeft in het pseudo-label. Deze aanpak maakt FixMatch efficiënt en relatief robuust, en bovendien minder afhankelijk van uitgebreid afgestelde parameters. De eenvoud van de methode betekent echter niet per se dat het altijd de beste prestaties levert, zeker niet wanneer het vertrouwen van het model in

vroege training nog beperkt is.

Eigenschap	FixMatch	MixMatch
Kernidee	Pseudo-labeling + consistency	consistency + data mixing
Complexiteit	Eenvoudig	Complexer
Afhankelijkheid van tuning	Laag	Hoog
Robuustheid	Hoog	Matig
Afhankelijkheid van augmentaties	Ja (zwak + sterk)	Ja (mixing + sharpening)
Prestaties op kleine datasets	Goed	Afhankelijk van tuning

**Tabel 2.2:** Tabel met een simpele vergelijking tussen de onderzochte SSL-modellen binnen dit onderzoek.

### 2.5.3. Self-supervised learning

BYOL, MoCo en SimCLR vormen drie invloedrijke methoden binnen Self-SL die elk proberen om representaties te leren zonder gebruik te maken van gelabelde data. Alle drie maken gebruik van augmentaties en encoderstructuren om betekenisvolle visuele representaties te extraheren, maar verschillen fundamenteel in hoe zij het leerproces structureren – vooral met betrekking tot het gebruik van contrastieve loss en negatieve voorbeelden.

SimCLR is gebaseerd op contrastieve learning en vereist grote batch sizes om voldoende negatieve paren te genereren, wat in praktijk computationeel intensief kan zijn. Het model leert door representaties van verschillende augmentaties van hetzelfde beeld dichterbij elkaar te brengen, en andere beelden juist uit elkaar te duwen. Deze aanpak is krachtig op grote, diverse datasets, maar kan kwetsbaar zijn in domeinen met beperkte data of lage variatie, zoals sonarbeelden.

MoCo probeert de batch size-beperkingen van SimCLR te omzeilen door gebruik te maken van een momentum-geüpdatete geheugenbank die een groot aantal negatieve representaties opslaat. Hierdoor is MoCo efficiënter in resourcegebruik, maar het introduceert ook extra architecturale complexiteit en is gevoelig voor de consistentie van representaties over tijd – iets wat uitdagend kan zijn bij sonardata, waar de semantische grenzen tussen objecten niet altijd scherp zijn.

BYOL doorbreekt het contrastieve paradigma volledig door géén negatieve voorbeelden te gebruiken. In plaats daarvan traint het een online encoder om consistente representaties te leren ten opzichte van een langzaam geüpdatete target encoder. Deze aanpak is conceptueel eenvoudiger en robuuster, vooral in omge-

vingen met weinig visuele diversiteit of waar het definiëren van negatieve paren onnatuurlijk is. **BYOL** vereist bovendien minder computationele resources en presteert stabiel bij kleinere datasets en **batch sizes** – eigenschappen die relevant zijn voor toepassing op sonardata, waar gelimiteerde data beschikbaar is en objecten visueel subtiel kunnen zijn.

Eigenschap	SimCLR	MoCo	BYOL
Gebruik van negatieve paren?	Ja	Ja	Nee
<b>Batch size</b> vereisten	Hoog	Laag	Laag
Architectuurcomplexiteit	Gemiddeld	Hoog (met geheugenbank)	Gematigd
Stabiliteit bij kleine datasets	Laag	Gematigd	Hoog
Prestaties bij weinig variatie	Matig	Matig	Goed

**Tabel 2.3:** Tabel met een simpele vergelijking tussen de onderzochte **Self-SL**-modellen binnen dit onderzoek.

## 2.6. Overzicht van bestaande datasets en annotatietechnieken

### 2.6.1. Beschikbaarheid van publieke datasets

Zoals al eerder vermeld, is er voor het trainen van een objectdetectiemodel een substantiële hoeveelheid gelabelde data nodig. Tegenwoordig worden veel datasets – zo goed als volledig geprepareerd voor het trainen van ML-modellen – online aangeboden onder een vrije licentie. Dit zorgt ervoor dat de datasets opnieuw gepubliceerd en hergebruikt mogen worden met weinig tot geen beperkingen. Het concept van *Open Data* wordt de laatste jaren steeds populairder, aangezien onderzoek zich hoe langer hoe meer ook naar het internet verplaatst. (Murray-Rust, 2008)

Platformen zoals [Kaggle](#) of [Hugging Face](#) hebben zich gespecialiseerd in het verspreiden van deze open datasets en kunnen rekenen op een hele community van gebruikers om deze datasets te uploaden naar hun platformen. Ook verschillende overheidsorganisaties en onderzoeksinstituten maken veel van hun data openbaar. Voorbeelden hiervan zijn [data.europa.eu](#) voor het dataplatform van de Europese Unie, [data.gov.be](#) voor het Belgische Data Portaal en [data.gov](#) voor open data van de Amerikaanse overheid.

Voor datasets over niche onderwerpen zijn de mogelijkheden vaak beperkter. Dit heeft verschillende redenen: eerst en vooral zijn er minder mensen met deze onderwerpen bezig, wat de kans automatisch kleiner maakt dat ze (grote) datasets gaan verzamelen en deze publiek beschikbaar maken. Ten tweede brengt een dataset samenstellen een aanzienlijke kost met zich mee, zeker wanneer deze *from scratch* gemaakt moet worden en men zich bijvoorbeeld niet kan baseren op reeds bestaande datasets. Ten derde is het vaak zo dat deze zeer gespecialiseerde datasets confidencieel moeten blijven, aangezien het bijvoorbeeld gaat over gevoelige data of data over een technologie die nog volop in ontwikkeling is.

### 2.6.2. Problemen bij sonardatasets

Sonardata leidt aan alle drie deze problemen. Sonardata van de zeebodem moet namelijk met een gespecialiseerde AUV verzameld worden. Anders dan bij een Remotely Operated Vehicle (ROV) kan deze volledig autonoom opereren en is daarom zeer geschikt om sonaropnames te maken. Weinig bedrijven hebben echter een nuttige reden om de kost hiervan te kunnen rechtvaardigen. Daarnaast is er namelijk nog een hele crew en een schip nodig om dergelijk onderzoek uit te voeren. Toch zijn er organisaties die deze nood hebben aan deze data. Specifiek gaat dit dan over onderzoeksinstellingen die bepaalde dingen op de zeebodem onderzoeken. Maar ook Defensie en Marine kunnen zulke data enorm goed gebruiken voor veiligheidsdoeleinden. AUV's worden in deze context gebruikt om bijvoorbeeld zee-mijnen op te sporen, in kaart te brengen en onschadelijk te maken.

Ook worden ze gebruikt om te patrouilleren: ze kunnen hierbij kritieke infrastructuur zoals haveningangen en zeekabels bewaken en zorgen dat deze veilig en onbeschadigd blijven. Dit brengt natuurlijk onmiddellijk het derde probleem met zich mee. Alle data die door Defensie of Marine is verzameld, is automatisch confidencieel en mag onder geen enkele voorwaarde gebruikt – laat staan gepubliceerd – worden voor andere doeleinden. Data van de zeebodem – en van mijnen of kritieke infrastructuur – is namelijk van onschatbare waarde voor een potentiële vijand. (Aubard, Madureira e.a., 2024)

### 2.6.3. Mogelijke oplossingen

Deels omwille van de niche, deels door de hoge kosten van de apparatuur en deels door de hoge graad van gegevensbescherming is er dus nagenoeg geen – kwalitatieve – sonardata vrij beschikbaar. Dit is echter ook nadelig voor verdere ontwikkeling binnen het domein. Daarom zijn er – nog maar heel recent – enkele initiatieven opgekomen om dit alsmaar groter wordend probleem te proberen oplossen.

In een artikel van Aubard, Madureira e.a. (2024) gepubliceerd in IEEE Journal of Oceanic Engineering wordt het probleem aangehaald en worden enkele oplossingen

aangereikt om om te gaan met de schaarste. Daarnaast wordt een vergelijking gemaakt van (bijna) alle sonardatasets – die overigens nog maar heel recent publiek beschikbaar zijn. Er worden in het artikel verschillende soorten datasets voor verschillende doeleinden gepresenteerd. Zo is er een aanbod van verschillende soorten sonar, verschillende formaten van data, verschillende hoeveelheden data, verschillende types objecten in de data en data voor verschillende doeleinden (zoals classificatie, objectdetectie, segmentatie, ...).

In dit onderzoek wordt de focus gelegd op objectdetectie. Hierdoor vallen er al veel datasets af. Er blijven nog zo'n 5 datasets over die mogelijk gebruikt kunnen worden.

Dataset	Sonar	Hoeveelheid	Object labels	Jaar
UATD	FLS	9200	Banden, Kooien, ROVs	2022
SSS for Mine Detection	SSS	1170	Mijnen	2024
SWDD	SSS	7904	Muren	2024
SubPipe	SSS	10030	Pijpleidingen	2024
UXO	FLS	74437	Blindgangers	2024

**Tabel 2.4:** Tabel van mogelijk bruikbare datasets voor dit onderzoek.

Merk op dat al deze datasets nog niet lang geleden zijn gepubliceerd, wat er op wijst dat er slechts heel recent aandacht wordt besteed aan dit specifieke probleem. Op zich zijn al deze datasets van dermate kwaliteit dat ze kunnen dienen om een objectdetectiemodel te trainen. Echter vallen bepaalde datasets buiten de scope van dit onderzoek, zoals de SWDD dataset. Deze dataset is gemaakt voor de ontwikkeling van het ROSAR-framework in een paper van Aubard, Antal, Madureira e.a. (2024). Ze is echter minder geschikt om te gebruiken in dit onderzoek, aangezien het namelijk sonarbeelden die gemaakt zijn door een AUV alle havenmuren van de haven van Porto de Leixões in Portugal te laten volgen en deze af te scannen bevat. (Aubard, Antal, Madureira e.a., 2024)

Ook de SubPipe-dataset biedt geen echte meerwaarde voor dit onderzoek. Ze bevat namelijk beelden voor de inspectie van pijpleidingen. Echter bewijst deze dataset wel dat er door de initiatieven van de afgelopen jaren datasets ter beschikking gesteld zijn met voldoende data om een robuust model mee te trainen. SubPipe bevat ongeveer 80GB aan data, verdeeld in data voor SLAM, objectdetectie en segmentatie. Ook worden er (normale) foto's in hoge resolutie (gemaakt met een GoPro Hero 10) en annotatie/metadatum in verschillende formaten meegeleverd. (Álvarez-Tuñón e.a., 2024)

Zo blijven er nog 3 datasets over die mogelijks geschikt zijn om in dit onderzoek te gebruiken.

### UATD

De Underwater Acoustic Target Detection-dataset bevat meer dan 9000 **Multibeam Forward-Looking Sonar (MFLS)**-sonarbeelden. Ze bevat de *raw* sonar data met annotatie. Er komen 10 verschillende categorieën van *target*-objecten in voor, waaronder kooien, banden en cilinders. De data is – in tegenstelling tot sommige andere datasets – verzameld door een **AUV** in meren en plassen. De data komt met andere woorden uit “de echte wereld”, wat voordelig is voor de training van modellen die in *real-world*-scenario's gebruikt zullen worden. (Xie e.a., 2022)



**Figuur 2.6:** Overzicht van objecten in de UATD-dataset. (Xie e.a., 2022)

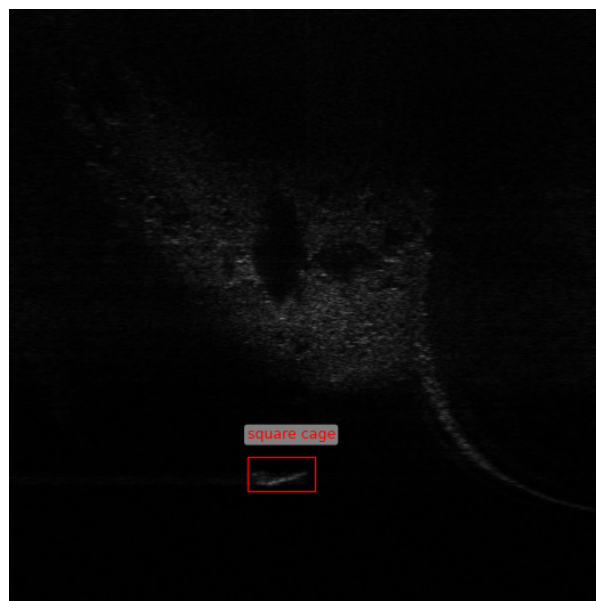
De dataset wordt verspreid in een figshare-repository onder de CC BY 4.0 licentie. Deze licentie laat de gebruiker toe om het materiaal te delen en te bewerken, ook voor commerciële doeleinden. Buiten een naamsvermelding van de maker, een link naar de licentie en een vermelding of het werk al dan niet veranderd is, zijn er geen verdere restricties. De dataset wordt als ZIP-archief aangeboden. Gecomprimeerd is deze ongeveer 4,47 GB groot. Uitgepakt bevat ze drie aparte datasets en OpenSLT, de annotatietool die gebruikt werd, telkens opnieuw apart verpakt als ZIP-archief. De drie datasets zijn als volgt verdeeld: twee testsets – UATD\_Test\_1 en UATD\_Test\_2 – en één trainingsset – UATD\_Training. Elk archief – buiten die van de annotatietool natuurlijk – bevat twee mappen: *annotations* en *images*. De map *annotations* bevat de annotaties en de map *images* bevat de sonarbeelden. (Jian & Kaibing, 2022)

Dataset	Aantal bestanden	Gecomprimeerde grootte	Ware grootte
UATD_Test_1	800	421.9 MB	3.3 GB
UATD_Test_2	800	424.6 MB	3.3 GB
UATD_Training	7600	3.9 GB	25.8 GB

**Tabel 2.5:** Tabel van eigenschappen van de datasets binnen de UATD-dataset. (Jian & Kaibing, 2022)

De bestanden volgen een vast benamingsschema zodat de beelden en de annotaties makkelijk aan elkaar gekoppeld kunnen worden. In elke dataset zijn de bestanden genummerd van 00001 tot aan het aantal bestanden in de dataset (bv. 00800 voor elke testset). Het beeld en de overeenkomstige annotatie hebben hetzelfde nummer. Zo zijn de paden voor het tweede beeld bijvoorbeeld:

- **Beeld:** UATD/UATD\_Test\_1/images/00003.bmp
- **Annotatie:** UATD/UATD\_Test\_1/annotations/00003.xml



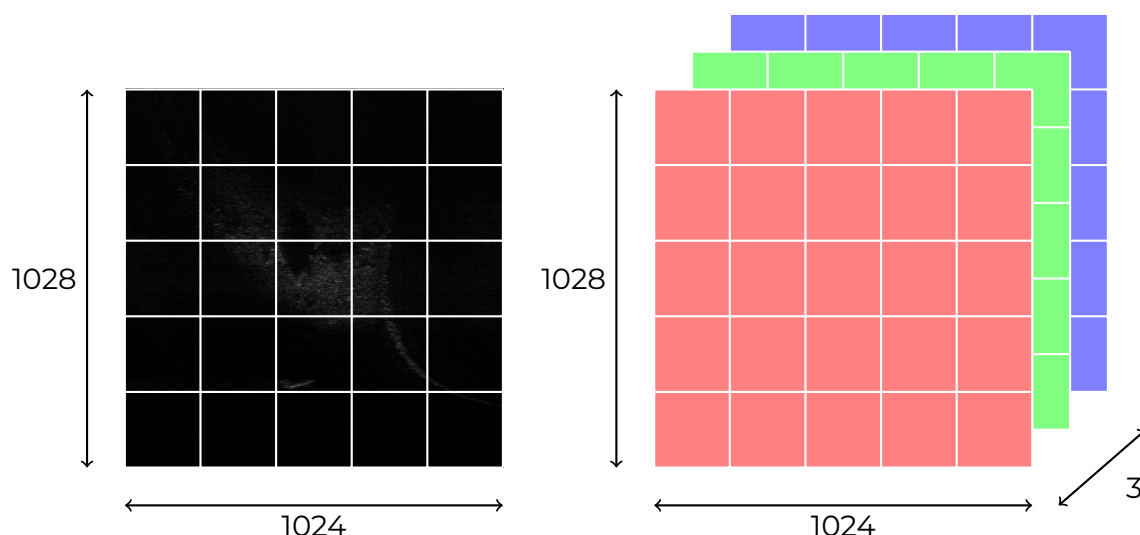
**Figuur 2.7:** Voorbeeld van een sonarbeeld binnen de UATD-dataset met de objecten aangeduid met **bounding boxes** (afbeelding UATD\_Test\_1/00003). (Xie e.a., 2022)

De beelden zijn opgeslagen als 3-kanaals ongecomprimeerde BMP-bestanden met een bitdiepte van 8 bits (één byte). Dit is één van de eenvoudigste bestandsformaten om een foto op te slaan. Het bestand bevat namelijk een lijst van alle pixels, waarbij voor elke pixel het kleur wordt bijgehouden. Het kleur wordt in dit geval dus opgeslagen in 8 bits. Dit betekent dat het kleur van elke pixel bepaald wordt door een waarde tussen 0 en 255. Aangezien er drie kanalen zijn (RGB: Rood, Groen

en Blauw), wordt elke pixel dus beschreven door  $8 \times 3 = 24$  bits of 3 bytes. Als de afmetingen van de afbeelding gekend zijn, is het makkelijk om de grootte van de afbeelding te berekenen met behulp van volgende formule:

$$\text{Grootte in Bytes} = \frac{\text{breedte} \times \text{hoogte} \times \#\text{kanalen} \times \text{bitdiepte}}{8}$$

Naast enkele tientallen bytes aan header- en fotodata is dit de werkelijke grootte van een BMP-bestand. Het nadeel is echter dat zo'n bestand heel snel heel groot wordt, aangezien BMP meestal gebruikt wordt zonder compressie. Anderzijds is zo'n verzameling van BMP-bestanden dan weer heel goed te comprimeren in bijvoorbeeld een ZIP-bestand. Zo kon de UATD-dataset van ongeveer 32,8 GB naar ongeveer 4,75 GB gecomprimeerd worden<sup>1</sup>, wat op een compressiegraad van ongeveer 6.8 neerkomt. (Bourke, 1998)



**Figuur 2.8:** Voorbeeld van de structuur van een bitmap-afbeelding zoals BMP.

Per afbeelding is er steeds een overeenkomstig XML-bestand dat de annotatie bevat. De annotatie volgt het Pascal VOC-formaat. Dit annotatieformaat werd origineel ontwikkeld voor de Visual Object Challenge. Deze challenge is een benchmark voor classificatie en objectdetectie die sinds 2005 jaarlijks georganiseerd wordt. Ondertussen is het formaat uitgegroeid tot een veelgebruikte standaard voor de annotatie van visuele datasets voor bijvoorbeeld objectdetectie. Door het gebruik van XML is het formaat makkelijk leesbaar. Echter maken heel weinig modellen rechtstreeks gebruik van het Pascal VOC-formaat, waardoor de annotatie meestal moet omgezet worden naar een ander formaat. (Everingham e.a., 2009)

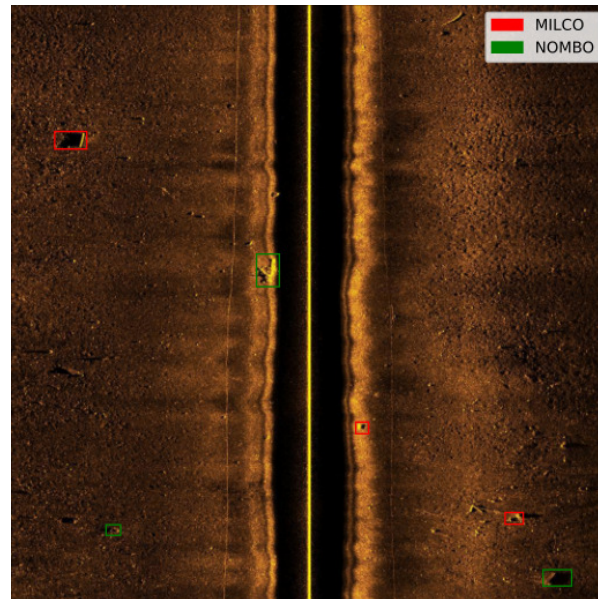
<sup>1</sup>Dit gaat over de som van de drie ZIP-bestanden die nog eens gezipd in de UATD-dataset zaten. De drie bestanden zijn dus nog eens gecomprimeerd, wat de grootte van het origineel gedownloadde bestand op 4,47 GB bracht.

```
1 <annotation>
2   <sonar>
3     <range>24.9909</range>
4     <azimuth>120</azimuth>
5     <elevation>12</elevation>
6     <soundspeed>1498.1</soundspeed>
7     <frequency>1200k</frequency>
8   </sonar>
9   <file>
10    <folder>UATD_Test_1</folder>
11    <filename>00004</filename>
12  </file>
13  <size>
14    <width>1024</width>
15    <height>1257</height>
16    <channel>3</channel>
17  </size>
18  <object>
19    <name>rov</name>
20    <bndbox>
21      <xmin>574</xmin>
22      <ymin>880</ymin>
23      <xmax>618</xmax>
24      <ymin>909</ymin>
25    </bndbox>
26  </object>
27 </annotation>
```

**Codefragment 2.1:** Voorbeeld van een XML-bestand met annotatie voor objectdetectie in het PASCAL VOC-formaat (annotatie van UATD\_Test\_1/00004). (Xie e.a., 2022)

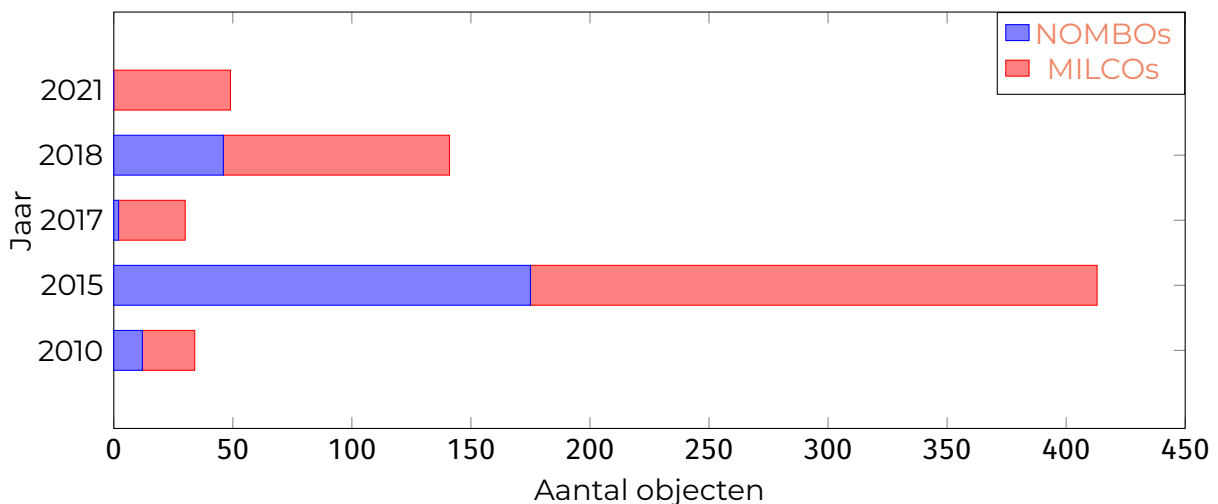
### SSS for Mine Detection

Een andere mogelijke kandidaat is de *SSS for Mine Detection*-dataset. Deze dataset bevat 1170 *real-world Side-Scan Sonar (SSS)* sonarbeelden van de Portugese kust gemaakt door een gespecialiseerde AUV. Een dataset van deze grootte en kwaliteit verzamelen is geen sinecure, daarom is deze dataset verzameld in samenwerking met de Portugese marine. Specifiek werkten de onderzoekers gedurende verschillende jaren samen met *Destacamento de Mergulhadores Sapadores (DMS 3)*. Dit team is verantwoordelijk voor alles wat met mijnen in zee te maken heeft. (Pessanha Santos e.a., 2024)



**Figuur 2.9:** Voorbeeld van een sonarbeeld uit de *SSS for Mine Detection*-dataset met de objecten aangeduid met *bounding boxes* (afbeelding *SSS for Mine Detection/2015/0001\_2015*). (Pessanha Santos & Moura, 2024)

Deze dataset is geschikt voor verschillende soorten ML-gerichte taken, waaronder objectdetectie, classificatie en segmentatie. De objecten in de dataset zijn geannoteerd met *bounding boxes* om verschillende objecten aan te duiden. Deze vallen onder te verdelen in twee klassen: *MIne-Like COntacts (MILCOs)* en *NOOn-Mine-like BOttom Objects (NOMBOs)*. Kortom: alles wat een mijn kan zijn en alles wat geen mijn kan zijn.



**Figuur 2.10:** Grafiek van het aantal waargenomen objecten per jaar in de *SSS for Mine Data*-dataset. (Pessanha Santos e.a., 2024)

Ook deze dataset wordt verspreid in een figshare-repository onder de CC BY 4.0 licentie. Opnieuw wordt een ZIP-archief van ongeveer 584 MB aangeboden. Dit

ZIP-archief bevat opnieuw andere ZIP's. Deze hebben de naam van één van de jaren waarvan er data beschikbaar is en bevatten dit dan ook. Daarnaast is er nog een ZIP-bestand met de gewichten van een getraind v4-model en de code hiervoor. Dit is echter niet nuttig voor dit onderzoek en zal dus niet gebruikt worden.

In de mappen verdeeld per jaar zitten zowel de beelden als de annotaties (niet gescheiden). De beelden zijn JPG-bestanden. Het lijkt erop dat er twee verschillende resoluties gebruikt zijn voor de afbeeldingen, namelijk 416 x 416 en 1024 x 1024. De afbeeldingen volgen een vast benamingsschema: elke afbeelding bevat een oplopend nummer beginnend van 0001, daarna een underscore en dan het jaar waarin de afbeelding is gemaakt. Alles samen is dat dus bijvoorbeeld: 0001\_2015.jpg. De bestanden met annotatie staan in dezelfde map en hebben dezelfde naam als hun overeenkomstige afbeeldingen. Het enige verschil is dat zij een .txt-extensie hebben. Dit tekstbestand bevat de annotatie in YOLO-formaat. (Pessanha Santos & Moura, 2024)

---

```

1 0 0.10009765625 0.2265625 0.0537109375 0.029296875
2 1 0.43017578125 0.4443359375 0.0380859375 0.0546875
3 0 0.587890625 0.7080078125 0.021484375 0.01953125
4 0 0.84228515625 0.859375 0.0302734375 0.01953125
5 1 0.17138671875 0.87890625 0.0244140625 0.017578125
6 1 0.91455078125 0.958984375 0.0478515625 0.02734375

```

---

**Codefragment 2.2:** Voorbeeld van een TXT-bestand met annotatie voor objectdetectie in het YOLO-formaat (annotatie van SSS for Mine Detection/2015/0001\_2015). (Pessanha Santos & Moura, 2024)

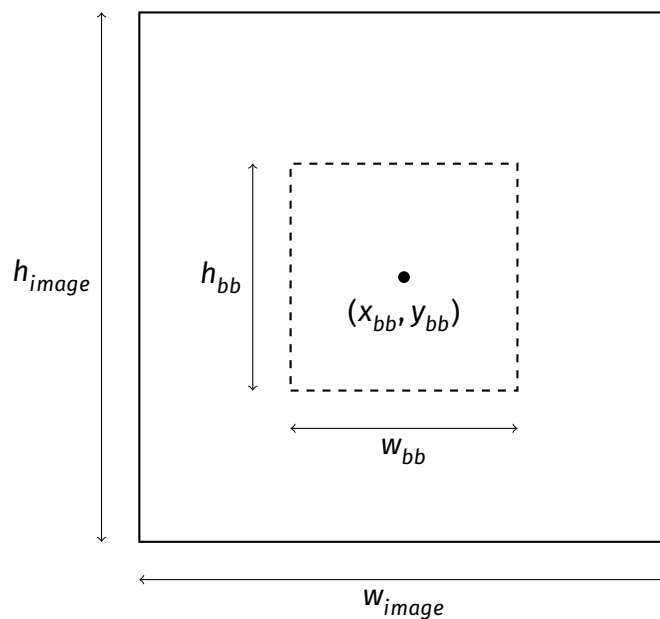
Eigenlijk is dit niks meer dan een CSV-bestand waarbij de separator een spatie is. In getabelleerde vorm is de data iets overzichtelijker.

Klasse	x-coördinaat	y-coördinaat	Hoogte	Breedte
0	0.10009765625	0.2265625	0.0537109375	0.029296875
1	0.43017578125	0.4443359375	0.0380859375	0.0546875
0	0.587890625	0.7080078125	0.021484375	0.01953125
0	0.84228515625	0.859375	0.0302734375	0.01953125
1	0.17138671875	0.87890625	0.0244140625	0.017578125
1	0.91455078125	0.958984375	0.0478515625	0.02734375

**Tabel 2.6:** Tabel met annotatie voor objectdetectie in het YOLO-formaat (annotatie van SSS for Mine Detection/2015/0001\_2015). (Pessanha Santos & Moura, 2024)

Merk op dat de klasse 0 of 1 is. Uit de paper van Pessanha Santos e.a. (2024) blijkt dat 0 een MILCO is en 1 een NOMBO. De volgende vier kolommen stellen een bounding box voor. Er bestaan verschillende formaten om zo'n bounding box voor te stellen. Dit formaat gebruikt één coördinaat  $((x, y))$  die het midden van de bounding box voorstelt, de hoogte en de breedte.

Men zou verwachten dat deze waarden gegeven zijn in pixels. Dat brengt echter een probleem met zich mee wanneer de afbeelding geschaald wordt. Dan moeten de pixelwaarden telkens herberekent worden om zo de bounding box op de juiste plaats op de afbeelding te laten vallen. Om dit op te lossen worden deze waarden – zoals hier – meestal als verhoudingen uitgedrukt. Zo is de breedte bijvoorbeeld de breedte in pixels gedeeld door de breedte van de afbeelding.



**Figuur 2.11:** Structuur van een bounding box (gebaseerd op een figuur van Pessanha Santos e.a. (2024))

Als de afbeelding geschaald wordt, schaalt de bounding box mee en kan men aan de hand van een relatief eenvoudige formule terug de pixelwaarde berekenen zonder complexe transformaties op de bounding box te moeten gaan toepassen. De formule om de absolute pixelwaarden om te zetten naar relatieve verhoudingen is de volgende:

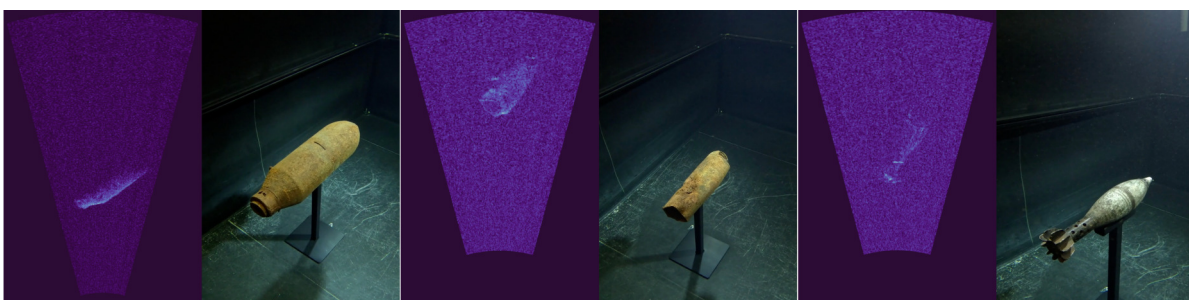
$$(x, y, w, h) = \left( \frac{x_{bb}}{w_{image}}, \frac{y_{bb}}{h_{image}}, \frac{w_{bb}}{w_{image}}, \frac{h_{bb}}{h_{image}} \right)$$

Naast het verzamelen van de dataset trainden de onderzoekers ook een objectdetectiemodel – namelijk YOLOv4 – op deze data om de kwaliteit ervan te testen. De configuratie voor het trainingsproces werd speciaal aangepast om de performance van dit model te verhogen. De **batch size** werd ingesteld op 64 en werden gedurende de training opgesplitst in 16 **mini-batches** van elk 4 afbeeldingen, dit om het geheugengebruik tijdens het trainingsproces te optimaliseren.

Het maximum aantal **batches** werd ingesteld op 6000. Ook werd de **learning rate** van het model op kritieke punten aangepast, namelijk na het verwerken van 4800 en 5400 **batches**, dit om de convergentie te verhogen. Ook pasten de onderzoekers transfer learning toe door de gewichten van het model te initialiseren met gewichten van de **Common Objects in Context (COCO)**-dataset. Na het model te trainen op de volledige dataset van 1170 afbeeldingen, behaalde deze een **IoU** van 60% en een **mAP** van 75%. Ook werd een **precision** van 82% en een **recall** van 64% behaald. (Pessanha Santos e.a., 2024)

### UXO

Ten slotte is er de **UXO**-dataset. **UXO** is de afkorting voor **Unexploded Ordnance**, de Engelse benaming voor **blindgangers**. Dit is meteen ook de inhoud van deze dataset. Ze bevat namelijk 74 437 afbeeldingen van **blindgangers** en is daarmee de grootste dataset die in dit onderzoek voorkomt. Het doel van deze dataset is om een validatieset te zijn en ze is samengesteld om onderzoek naar ontminning in plassen, meren, zeeën en oceanen vooruit te helpen. Zoals al eerder vermeld is dit onderwerp zeer gevoelig en wordt er meestal (lees: bijna altijd) voor gekozen om deze datasets zo confidentieel mogelijk te houden. Daarom bevat deze dataset ook geen *real-world*-afbeeldingen. In plaats daarvan zijn beelden in een gecontroleerde, experimentele omgeving gemaakt. (Dahn, Bande Firvida e.a., 2024)



**Figuur 2.12:** . Voorbeeld van sonarbeelden en overeenkomstige afbeeldingen van verschillende type **blindgangers** in de UXO-dataset. (Dahn, Bande Firvida e.a., 2024)

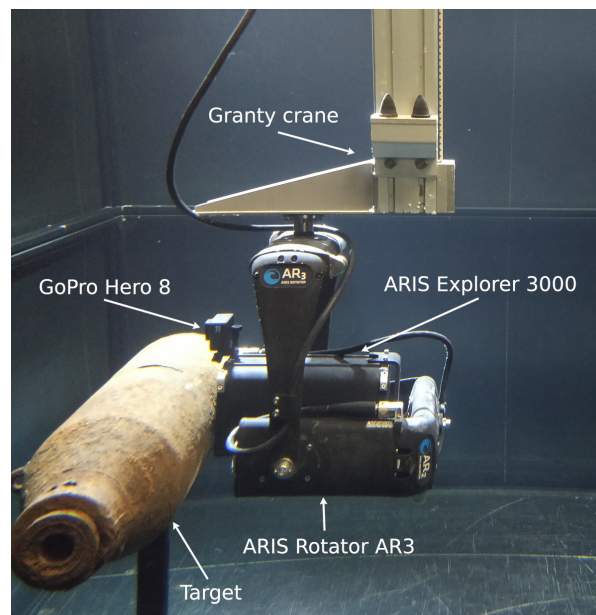
Om deze dataset samen te stellen, hebben de onderzoekers een volledig gecontroleerde testopstelling gemaakt bij het **Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)**, het Duits onderzoekscentrum voor artificiële intelligentie in Bremen. De dataset is gemaakt met een ARIS Explorer 3000 sonarmodule (voor de

sonarbeelden) en een GoPro Hero 8 (voor de bijhorende 5,3K UHD afbeeldingen). Deze twee modules werden op een (de ARIS Rotator AR3) gemonteerd die vastzat aan een op maat gemaakte **portaalkraan**. Deze kraan kon vrij in de *xyz*-assen bewegen en kon verschillende voorgeprogrammeerde banen heel precies volgen.



**Figuur 2.13:** . Afbeelding van de ARIS Explorer 3000 sonarmodule. (soundmetrics.com, [g.d.](#))

Om de omgeving van de zee zo goed mogelijk te kunnen nabootsen, bouwden de onderzoekers een bassin gevuld met 20 000 liter zoetwater. Hierin plaatsten ze verschillende *targets*, de **blindgangers**. Deze waren aangeleverd door EGGERS Kampfmittelbergung GmbH, een bedrijf dat zich bezighoudt met het ontminnen van verschillende sites. Om de veiligheid te garanderen, werden de **blindgangers** voordien onschadelijk gemaakt. Er werden verschillende **blindgangers** gebruikt om zo variëteit in de dataset te garanderen: een normale bom van ongeveer 51 kg, een vervormde fosforbom en een mortier. (Dahn, Firvida e.a., [2024](#))



**Figuur 2.14:** . Afbeelding van de setup die gebruikt werd om data te verzamelen voor de UXO-dataset. (Dahn, Bande Firvida e.a., [2024](#))

De dataset is alles samen zo'n 94,7 GB groot. Ze is de meest uitgebreide die in dit onderzoek voorkomt, niet alleen qua grootte, maar ook qua inhoud. De dataset is onderverdeeld in verschillende mappen. De map `3d_models` bevat 3D-modellen van de **blindgangers**. De map `calibration` bevat dan weer de transformaties tussen de kraan en de sensors en de kalibraties van de GoPro. Het merendeel van de dataset zit in de map `recordings`. Deze bevat mappen per type **blindganger**. In deze mappen zitten dan telkens weer mappen met de datum en tijd van elk experiment. Deze mappen bevatten verschillende mappen en bestanden die de *core* van de data vormen. (Dahn, Bande Firvida e.a., 2024)

- **aris\_raw**: map die de *raw* sonarbeelden bevat in PGM formaat. Dit is een eenvoudig rasterafbeeldingsformaat (zoals BMP, cf. figuur 2.8) dat grijswaardenafbeeldingen opslaat. PGM-bestanden kunnen in een ASCII (tekst) of binaire (snellere) variant worden opgeslagen. Elke pixel heeft een intensiteitswaarde tussen 0 (zwart) en een maximumwaarde (meestal 255, wit), waardoor verschillende grijs tinten mogelijk zijn. De afbeelding bevat ook een header met het formaat, afmetingen en maximale intensiteit, gevolgd door de pixelgegevens. PGM wordt vaak gebruikt in beeldverwerking en wetenschappelijke toepassingen vanwege de eenvoud en brede compatibiliteit. (Poskanzer, 2016)
- **aris\_polar**: map die de poolgetransformeerde sonarbeelden bevat in PNG formaat.
- **gopro**: map die de beelden in JPG formaat bevat die gemaakt zijn door de GoPro.
- **labels**: map die annotatie van **bounding boxes** bevat in JSON formaat.
- **aris\_file\_meta.yaml**: YAML-bestand met de metadata van de sonar
- **aris\_frame\_meta.csv**: CSV-bestand met metadata voor elk sonarframe, inclusief **Pan-Tilt Unit (PTU)**-informatie.
- **gantry.csv**: CSV-bestand met posities van de **portaalkraan** bij elk sonarframe.
- **notes.txt**: TXT-bestand met korte beschrijving over experiment.



**Figuur 2.15:** . Preview van de setup waarmee de UXO-dataset is gemaakt in actie. (Dahn, Bande Firvida e.a., 2024)

# 3

## Methodologie

Dit onderzoek volgt een gestructureerde aanpak om **SSL** en **Self-SL** technieken voor objectdetectie in sonardata te implementeren en te evalueren. In deze methodologie wordt een onderverdeling gemaakt van de verschillende fasen in dit onderzoek. Hierbij wordt de basis gelegd voor de experimenten die zullen worden uitgevoerd in de proof of concept.

### 3.1. Data-acquisitie

Allereerst moet er een keuze gemaakt worden voor het gebruik van een dataset in het verdere verloop van dit onderzoek. Alle drie de datasets die in 2.6.3 besproken werden, maken het mogelijk om een objectdetectiemodel mee te trainen. Ze bieden namelijk allemaal annotaties van **bounding boxes** op de bijhorende afbeeldingen aan. Daarnaast zijn deze datasets makkelijk te verwerken, aangezien alle beelden in conventionele afbeeldingsformaten zijn opgeslagen (zoals PNG, JPG, BMP, PGM, ...). Toch is er – specifiek voor dit onderzoek – één dataset die geschikter is dan de anderen. De UATD-dataset is – misschien ietwat subjectief – uitgekozen om te gebruiken in de rest van dit onderzoek. Dit komt omdat ze bepaalde aspecten aanbiedt die de andere datasets niet hebben.

**SSS** for Mine Detection lijkt op het eerste zicht de perfecte dataset voor dit onderzoek. Het probleem is echter dat ze relatief klein is: ze bevat “slechts” 1170 afbeeldingen. Op het eerste zicht lijkt dit voldoende. Echter moet deze dataset nog opgesplitst worden in – ten minste – een trainingsset en een testset.<sup>1</sup> Ook komen er

<sup>1</sup>In een optimale situatie zou de data opgesplitst worden in drie sets: een trainingsset, een testset en een validatieset. Dit komt omdat de validatiedataset – hoewel ze niet gebruikt wordt om het model te trainen – gebruikt wordt om de parameters van het model te tunen. Dit kan leiden tot **overfitting**. Het is beter om als testset data te gebruiken dat het model nog niet gezien heeft. (Goodfellow e.a., 2016)

slechts 668 objecten voor in de dataset. Tot overmaat van ramp zijn deze ook zeer slecht verdeeld.

# objecten / beeld	# beelden
13	1
9	2
8	4
7	8
6	8
5	13
4	9
3	41
2	59
1	159
0	866

**Tabel 3.1:** Tabel met verdeling van objecten per afbeelding in de *SSS* for Mine Detection-dataset.

Er is één afbeelding met wel 13 objecten en 866 zonder ook maar één object. Door deze slechte verdeling en de beperkte hoeveelheid data in de dataset is ze dus weinig bruikbaar voor dit onderzoek.

Dit is een probleem waar de UXO-dataset absoluut niet mee kampt. Deze heeft dan echter weer andere problemen. De dataset is namelijk volledig samengesteld in een gecontroleerde testopstelling. Ze bevat dus geen *real-world*-data. Dit betekent echter ook dat bepaalde artefacten en afwijkingen typisch aan meren en zeeën niet in deze dataset voorkomen. De beelden zijn zodanig zuiver dat het hoogstwaarschijnlijk mogelijk zou zijn om de *blindgangers* te herkennen door te zoeken naar de groep helderste pixels of met een edge-detection algoritme. (Torre & Poggio, 1986)

Ook de grootte van de dataset is misschien iets te mooi om waar te zijn. De beelden in de dataset zijn namelijk geen onafhankelijke afbeeldingen, maar frames van een continue opname. Dit zorgt ervoor dat er (nagenoeg) geen verschil is tussen afbeelding  $n$  en afbeelding  $n+1$ . Als alle afbeeldingen na elkaar worden afgespeeld, ziet men een opname van een transformatie (rotatie, verschuiving, ...) van één van de *blindgangers*. Ten slotte staat er telkens maar één object op een afbeelding, wat multiple objectdetectie (meerdere objecten op één afbeelding herkennen) onmogelijk maakt.

## 3.2. Dataverdeling

In dit onderzoek zullen verschillende modellen getraind worden met verschillende leertechnieken. Daarom is het belangrijk om een duidelijk zicht te krijgen op hoe de gekozen dataset verdeeld moet worden zodat dit efficiënt en effectief kan gebeuren. Zoals vermeld zal er gebruik gemaakt worden van de UATD-dataset. Aangezien deze al opgesplitst is in drie subsets, zal de data als volgt verdeeld worden:

- UATD\_Training: trainingsset (7600 samples)
- UATD\_Test\_1: testset (800 samples)
- UATD\_Test\_2: validatieset (waar nodig/mogelijk) (800 samples)

De trainingsset bevat 7600 gelabelde samples. Om de hypothesen in dit onderzoek echter te kunnen testen, zal deze set om bepaalde modellen te trainen verder opgesplitst worden in een gelabelde en een ongelabelde set. Om de resultaten van SSL en Self-SL beter te kunnen evalueren, zal een volledig gesuperviseerd model getraind worden op subsets van de gelabelde data. Meer specifiek gaat dit om: 100% (7600 samples), 50% (3800 samples), 10% (760 samples), 5% (380 samples) en 1% (76 samples). In deze fase worden de overige ongelabelde samples gewoonweg niet gebruikt.

Omwille van de resource-intensiviteit om deze modellen te trainen en het feit dat dit “slechts” een proof of concept is, zullen de SSL- en Self-SL-modellen dit trainings-schema niet volgen. Ze worden getraind op de twee subsets die het interessantst zijn voor dit onderzoek. Een subset van 10% (760 samples) zal gebruikt worden om de praktische werking in een realistisch scenario aan te tonen. Daarnaast zal een subset van 5% (380 samples) gebruikt worden om een mogelijk scenario waar data zeer schaars is na te bootsen. De overige samples zullen telkens gebruikt worden als ongelabelde trainingsdata. De modellen worden in se dus telkens op de volledige trainingsset getraind, echter is de leertechniek – en dus de verhouding gelabeld/ongelabeld – voor elk model verschillend.

## 3.3. Modelselectie

### 3.3.1. Supervised: Faster R-CNN

Voor de gesuperviseerde baseline binnen dit onderzoek is gekozen voor Faster R-CNN. Deze keuze is gemotiveerd door de hoge nauwkeurigheid die het model biedt bij objectdetectie, in het bijzonder in situaties waar precisie belangrijker is dan snelheid. In tegenstelling tot de andere modellen: YOLO en SSD, die geoptimaliseerd zijn voor real-time toepassingen en een lagere latency, biedt Faster R-CNN doorgaans betere prestaties op het gebied van lokalisatie en classificatie. Deze eigenschappen zijn enorm waardevol in de context van sonardata, waar objecten

meestal subtiele kenmerken vertonen en nauwkeurige detectie essentieel is.

Sonardata is ruisachtig, bevat lage resolutiebeelden, en vertoont objecten met weinig visuele contrasten. Dit is zeer goed te zien in de dataset die voor dit onderzoek gebruikt wordt. Hierbij is het zelfs moeilijk om als mens zonder getraind oog objecten van elkaar te onderscheiden en te benoemen. In dergelijke gevallen is een model dat profiteert van een two-stage detectieproces, zoals Faster **R-CNN**, beter in staat om relevante objecten van achtergrondruis te onderscheiden. Dit two-stage proces – bestaande uit een **RPN** gevolgd door een classificatie- en regressiehead – stelt het model in staat om meer contextueel geïnformeerde en preciezere objectvoorspellingen te doen.

Bovendien wordt Faster **R-CNN** in de literatuur vaak gebruikt als objectdetectiemodel voor vergelijkbaar onderzoek, zo ook in de paper van Xie e.a. (2022), waar de dataset die ook voor dit onderzoek gebruikt wordt, praktisch onderzocht wordt. Dit maakt het model niet alleen geschikt voor de sonardataset zelf, maar ook vergelijkbaar met bestaande benchmarks, wat cruciaal is voor het evalueren van de effectiviteit van alternatieve leermethoden. In tegenstelling tot single-shot detectors zoals **SSD** en **YOLO**, die sneller zijn maar mogelijk minder goed generaliseren in **SSL** en **Self-SL**-settings, biedt Faster **R-CNN** een betere balans tussen generaliseerbaarheid en performance, vooral in domeinen met beperkte gelabelde data.

Ten slotte is Faster **R-CNN** flexibel en goed ondersteund in frameworks zoals PyTorch, waardoor het eenvoudig geïntegreerd kan worden in experimenten met **SSL** en **Self-SL**-technieken. Deze praktische overweging speelt eveneens een rol in de keuze voor dit model als fundament binnen dit onderzoek.

### 3.3.2. Semi-supervised: FixMatch

Voor het **SSL**-model is gekozen voor FixMatch, een methode die sterke prestaties laat zien in settings met beperkte gelabelde data. FixMatch combineert de simplicitéit van pseudo-labeling met de effectiviteit van consistency regularization. Binnen FixMatch worden deze twee benaderingen op een efficiënte manier geïntegreerd. In tegenstelling tot het traditionele pseudo-labeling, waarbij zwak gecontroleerde voorspellingen worden gebruikt als labels voor ongesuperviseerde data, legt FixMatch een threshold op: alleen voorspellingen met hoge waarschijnlijkheid worden gebruikt als pseudo-labels. Hierdoor vermindert het risico op het versterken van foutieve voorspellingen – een cruciaal voordeel binnen sonardata, waar objecten vaak vaag of slecht afgebakend zijn.

Een ander sterk punt van FixMatch is het gebruik van sterke en zwakke augmentaties binnen het consistency framework. Door hetzelfde ongesuperviseerde voor-

beeld onder lichte (zwakke) en zware (sterke) augmentaties aan te bieden, en het model te trainen om consistente voorspellingen te doen, wordt het model robuust gemaakt. In de context van sonardata – waar visuele variatie door ruis of reflecties kan voorkomen – helpt dit mechanisme het model om betekenisvolle representaties te leren. Dit is effectiever dan eenvoudige consistency regularization zonder augmentatievariatie, dat gevoeliger kan zijn voor overfitting aan irrelevante ruis in de data.

Hoewel MixMatch ook een krachtig SSL-framework is dat meerdere strategieën (waaronder mixup) combineert, vereist het een relatief complex trainingsproces en intensieve hyperparametertuning. FixMatch daarentegen biedt een eenvoudigere implementatie en minder afhankelijkheid van hyperparameters, wat het aantrekkelijk maakt voor toepassingen in domeinspecifieke data zoals sonar, waar experimenteren met veel configuraties moeilijk is door beperkte data en computationele resources.

Samenvattend biedt FixMatch een sterke combinatie van betrouwbaarheid, eenvoud en robuustheid. De methode is bijzonder geschikt voor domeinen met weinig gelabelde data, hoge data-variabiliteit en lage visuele resolutie, zoals bij sonarbeelden. Hierdoor vormt het een degelijke keuze voor dit onderzoek, zowel vanuit technisch perspectief als met het oog op reproduceerbaarheid en vergelijkbaarheid met vergelijkbare literatuur in SSL.

### 3.3.3. Self-supervised: BYOL pre-training

Voor het Self-SL-leerproces is gekozen voor BYOL als pretrainingmethode, met als doel een krachtige feature extractor (backbone) te trainen zonder gebruik te maken van gelabelde data. Deze backbone wordt vervolgens geïntegreerd in de supervised baseline, Faster R-CNN, om de prestaties bij objectdetectie in sonardata te verbeteren. BYOL is geselecteerd vanwege zijn robuuste mogelijkheid om representaties aan te leren zonder negatieve paren, wat het bijzonder geschikt maakt voor data met beperkte visuele variatie zoals sonarbeelden.

In tegenstelling tot methoden zoals SimCLR en MoCo, die sterk leunen op contrastieve loss en dus negatieve voorbeelden nodig hebben, leert BYOL door het afstemmen van een online netwerk op een target netwerk dat langzaam geüpdatet wordt via momentum. Dit maakt BYOL stabiel en effectiever bij kleinere batch sizes, een belangrijk voordeel in scenario's met beperkte computationele resources of datasets die niet rijk zijn aan semantische variatie, zoals bij sonar. Bovendien vermindert het ontbreken van negatieve paren het risico dat semantisch gelijkaardige objecten (zoals verschillende onderwaterstructuren) foutief als verschillend worden gemodelleerd.

**SimCLR**, hoewel krachtig in standaard visuele benchmarks, vereist grote **batch sizes** en geavanceerde augmentatiestrategieën om goed te presteren, wat in de praktijk minder haalbaar is bij domeineigen data zoals sonar. **MoCo** probeert dit te verhelpen met een memory bank, maar introduceert tegelijkertijd extra complexiteit, berust sterk op hyperparameters en vertrouwt nog steeds op de aanwezigheid van duidelijke negatieve voorbeelden – iets wat binnen sonar data niet vanzelfsprekend is. In vergelijking daarmee biedt **BYOL** een eenvoudiger trainingsproces met consistente resultaten, zelfs zonder zware tuning.

Tot slot heeft de literatuur aangetoond dat het pretrainen met **BYOL** en het finetunen voor een downstream taak – zoals **Faster R-CNN** in dit onderzoek – leidt tot betere generalisatie en snellere convergentie dan alternatieve **Self-SL**-methoden. In dit onderzoek speelt dit een cruciale rol, omdat het model zich moet kunnen aanpassen aan objecten die zelden voorkomen, slecht gedefinieerd zijn, of in onbekende vormen voorkomen. **BYOL** draagt bij aan het leren van semantisch rijke en transfereerbare representaties, die de prestaties van het uiteindelijke detectiemodel zullen verbeteren zonder afhankelijkheid van “dure” annotaties.

### 3.4. Resultaten en evaluatie

Uiteindelijk zullen de verschillende uitgevoerde experimenten worden geëvalueerd en met elkaar vergeleken. Dit zal op verschillende manieren gedaan worden. Specifiek gaat dit om kwantitatieve en kwalitatieve methoden. Kwantitatief zal vooral **Mean Average Precision (mAP)** gebruikt worden om objectdetectieprestaties te beoordelen. Echter is dit niet de enige kwantitatieve metriek die er toe doet. Naast de effectieve performantie van het model is het ook belangrijk om rekening te houden met de resources die nodig zijn om een bepaald model te trainen. Ook wordt een vergelijking gemaakt tussen dezelfde modellen die getraind zijn met een verschillende verdeling van gelabelde en ongelabelde data. Dit om de label-efficiëntie (Hoe goed presteert het model met een beperkte hoeveelheid gelabelde data?) te meten. Dit zal cijfermatig uitsluitsel geven over de effectiviteit van semi-supervised en self-supervised learning tegenover supervised learning.

Naast de kwantitatieve analyse wordt er ook een kwalitatieve analyse uitgevoerd. De focus ligt hierbij niet zozeer op de cijfers, maar eerder op de effectieve voorspellingen van de modellen. Hierbij zal sterk worden gebruik gemaakt van beeldmateriaal en de voorspellingen gemaakt door de modellen. Hoewel een score heel objectief uitsluitsel kan geven over de performantie van een model, zijn real-world voorbeelden nog steeds waardevol voor de uiteindelijke evaluatie. Een score houdt namelijk niet altijd (genoeg) rekening met dingen die voor de eindgebruiker belangrijk zijn en omgekeerd.

Uiteindelijk zullen enkele experts in sonaranalyse de bruikbaarheid van de resul-

taten beoordelen en aanbevelingen geven voor verdere verbeteringen. Ten slotte zullen de methodologie, resultaten en code worden gedocumenteerd om reproduceerbaarheid te waarborgen.

# 4

## Experimenten

In dit hoofdstuk worden de experimenten beschreven die uitgevoerd zijn om de effectiviteit van *SSL* en *Self-SL* strategieën te evalueren binnen het domein van objectdetectie op sonardata. Daarbij wordt een vergelijking gemaakt tussen een volledig supervised baseline (Faster *R-CNN*) en twee alternatieve benaderingen: Fix-Match als *SSL* methode en BYOL als *Self-SL* pretrainingstrategie. De experimenten zijn opgezet met verschillende hoeveelheden gelabelde data, om te onderzoeken in welke mate deze technieken het labelproces kunnen verlichten zonder een significant verlies in detectieperformantie. De opbouw, trainingsdetails en resultaten worden in de volgende paragrafen toegelicht.

### 4.1. Experimentele setup

#### 4.1.1. Deep learning frameworks en tools

Vandaag de dag bestaan er verschillende deep learning frameworks die de ontwikkeling en implementatie van neurale netwerken sterk vergemakkelijken. Enkele van de meest gebruikte zijn TensorFlow, PyTorch, JAX en Keras. TensorFlow en PyTorch zijn de twee dominante deep learning frameworks in het veld van artificiële intelligentie. Keras is een speciale uitzondering: het is namelijk een bibliotheek die compatibiliteit biedt tussen deze verschillende frameworks.

#### TensorFlow

TensorFlow (geïntroduceerd door Google in 2015) werd ontworpen met het oog op grootschalige productieomgevingen. Het framework biedt uitgebreide ondersteuning voor het deployen van modellen op verschillende platformen, zoals mobiele apparaten en webapplicaties, en beschikt over geavanceerde tools zoals TensorBoard voor visualisatie en TensorFlow Serving voor schaalbare modelinzet. TensorFlow's aanpak met statische computationele grafen (in vroege versies) maakte

het echter aanvankelijk minder intuïtief voor onderzoek en experimentatie, hoewel TensorFlow 2.x dit deels heeft verbeterd door de introductie van *eager execution*. TensorFlow biedt ook veruit de beste integratie met **Tensor Processing Units (TPUs)**. (Pang e.a., 2019)

### PyTorch

PyTorch (ontwikkeld door **Facebook AI Research (FAIR)** en uitgebracht in 2016) heeft daarentegen vanaf het begin sterk ingezet op gebruiksvriendelijkheid en flexibiliteit. PyTorch maakt gebruik van dynamische computationele grafen, wat betekent dat het netwerk direct kan worden aangepast tijdens de uitvoering. Dit maakt het debuggen eenvoudiger en laat onderzoekers sneller experimenteren met nieuwe architecturen en technieken. Bovendien sluit de programmeerstijl van PyTorch dichter aan bij standaard Python, wat de leercurve verlaagt en de ontwikkelsnelheid verhoogt. In de onderzoekswereld heeft PyTorch hierdoor snel populariteit gewonnen, en veel state-of-the-art modellen en papers publiceren tegenwoordig hun codebase standaard in PyTorch. (Imambi e.a., 2021)

Voor dit project is uiteindelijk gekozen voor PyTorch vanwege de flexibiliteit en transparantie die het biedt tijdens het ontwikkelen en experimenteren met objectdetectiemodellen. PyTorch biedt namelijk uitstekende integraties met moderne detectiebibliotheken zoals TorchVision, wat de implementatie en evaluatie van complexe modellen versnelt. Daarnaast is het makkelijk om gepretrainde detectiemodellen te importeren, waardoor in dit onderzoek gefocust kan worden op de nieuwe ontwikkelingen zonder de focus te moeten verschuiven naar complexe backbones. Ook de installatie, configuratie en setup van PyTorch zijn veel eenvoudiger dan bij TensorFlow. Deze eigenschappen maken PyTorch tot de meest geschikte keuze om de doelstellingen van dit onderzoek efficiënt en effectief te realiseren.

#### 4.1.2. Hardware en compute resources

Het trainen van deze objectdetectiemodellen is een intensief proces dat zowel op het vlak van data als computationele resources aanzienlijke vereisten stelt. Zoals uitgelegd in 2.2.1 combineert objectdetectie classificatie en lokalisatie: het model moet niet alleen herkennen wat er in een afbeelding aanwezig is, maar ook waar het zich bevindt via het voorspellen van **bounding boxes**. Tijdens het trainingsproces worden geannoteerde datasets gebruikt waarbij objecten met klasse-labels en coördinaten zijn aangeduid, zoals in COCO of Pascal **Visual Object Classes (VOC)**. Het model leert via een samengestelde **loss-functie**, die typisch zowel classificatiefouten als fouten in de **bounding boxes** omvat. Door middel van **backpropagation** worden de gewichten van het neurale netwerk aangepast om deze gecombineerde **loss-functie** te minimaliseren. Omdat objectdetectie vaak gebaseerd is op diepe **CNNs**, vereist het trainingsproces krachtige hardware.

### CPU / GPU / TPU

Omwille van deze vereisten, worden neurale netwerken vrijwel nooit getraind op **Central Processing Units (CPUs)**. Een CPU is namelijk zeer goed in het uitvoeren van *general-purpose* taken (besturingssysteem draaien, tekstverwerking, spreadsheets ...) op een sequentiële manier. Om hiervoor geoptimaliseerd te zijn, bevat een CPU relatief weinig cores. Dit varieert meestal tussen de 2 en de 64. Echter draaien deze cores op een hoge tot zeer hoge snelheid, meestal tussen de 2 en de 5 GHz. Het grote nadeel van CPUs is dat ze niet geschikt zijn voor taken die sterk geparallelliseerd worden uitgevoerd, zoals het trainen van een neuraal netwerk. Deze taak vergt namelijk een grote hoeveelheid aan matrix- en tensoroperaties, hetgeen efficiënt kan geparallelliseerd worden.

Voor deze use-case wordt dus meestal gebruik gemaakt van één of meerdere **Graphics Processing Units (GPUs)**. Deze zijn oorspronkelijk bedoeld om beelden te renderen om ze op een monitor weer te geven. Aangezien dit ook een taak is die in hoge mate geparallelliseerd kan worden, is een GPU ontworpen met (tien)duizenden kleinere cores in plaats van enkele grote cores. Deze cores hebben dan wel weer een lagere snelheid dan de CPU-cores. Doorheen de jaren worden GPUs steeds meer gebruikt om parallelliseerbare taken uit te voeren. Naast de training van neurale netwerken gaat dit bijvoorbeeld ook om gaming, 3D-rendering en het minen van cryptomunten zoals Bitcoin. (Anish Dev, 2014)

Echter is een GPU niet speciaal geoptimaliseerd om neurale netwerken te trainen. Dit probleem wordt opgelost door de TPU. Dit is een **Application Specific Integrated Circuit (ASIC)** die speciaal ontworpen is door Google voor het trainen van neurale netwerken. De TPU is geoptimaliseerd om matrixvermenigvuldigingen en andere tensoroperaties uit te voeren. (Jouppi e.a., 2017) Het nadeel van de TPU is dat de compatibiliteit met third-party frameworks zoals PyTorch beperkt is. TPUs werken het beste met TensorFlow, het deep learning-framework van Google zelf. (Y. E. Wang e.a., 2019)

Afhankelijk van modelcomplexiteit, datasetgrootte, batchgrootte en de gebruikte optimalisatie-instellingen kan trainingsduur variëren van enkele uren tot meerdere dagen. Daarom wordt in veel gevallen – en ook in dit onderzoek – gebruik gemaakt van vooraf getrainde modellen (pretrained backbones) als uitgangspunt, wat de trainingstijd aanzienlijk kan verkorten en – vooral bij kleinere datasets – betere prestaties oplevert.

### Toegang tot resources

Er zijn enkele mogelijkheden om toegang te krijgen tot GPUs voor het trainen van deep learning-modellen. Zo is het mogelijk om via **Google Colab** gratis toegang te krijgen tot een NVIDIA Tesla T4 GPU met 16 GB aan videogeheugen en 2560 CUDA-

cores. (TechPowerUp, [g.d.-b](#)) Ondanks dat deze GPU voor het eerst op de markt is gebracht in 2018, is ze nog steeds bruikbaar en nuttig voor AI-workloads. Ook biedt Google via Colab een TPUV2 aan met 8 cores. Een nadeel is wel dat deze resources slechts beperkt toegankelijk zijn afhankelijk van de drukte en hoe zwaar de resource wordt belast. Hierdoor is Google Colab minder geschikt om grote, complexe modellen te trainen. Een ander nadeel is de toegang tot de benodigde data.

Op Google Colab is er telkens een virtuele schijf voorzien van zo'n 40-60 GB. Echter wordt deze volledig verwijderd op het moment dat de runtime afgesloten wordt. De data wordt met andere woorden niet persistent opgeslagen. Datasets telkens opnieuw downloaden en voorbereiden verspilt onnodig tijd en resources. Ook is het mogelijk om Colab te koppelen aan Google Drive, maar ook daar is de opslagruimte beperkt (15 GB gratis) en de leesnelheid is te traag om er trainingsdata op te slaan. Deze problemen worden grotendeels verholpen wanneer er gekozen wordt om een dedicated GPU-server te huren in de cloud, maar dit doet de kosten natuurlijk oplopen, zeker wanneer er een groot en complex model getraind moet worden.

In dit onderzoek werd er daarom gekozen om de modellen te trainen op een dedicated trainingsserver van [Exail Robotics Belgium](#). Deze server is uitgerust met een Intel Core i7-11700K processor met 8 cores, 64 GB DDR4 RAM en een NVIDIA RTX A5000 GPU met 24 GB aan videogeheugen en 8192 CUDA-cores. (TechPowerUp, [g.d.-a](#))

### 4.1.3. Optimalisatietechnieken

Doorheen de ontwikkeling van de verschillende modellen in dit onderzoek zal er gebruik gemaakt worden van een reeks optimalisatietechnieken om – onder andere – de performantie van de modellen te verbeteren, efficiënter met resources om te springen of sneller te trainen. Hieronder volgt een korte uitleg van enkele optimalisatietechnieken toegepast in dit onderzoek.

#### Early stopping

Een belangrijke optimalisatie binnen deep learning is het tegengaan van *overfitting*. Een mogelijke oorzaak hiervan is dat er te lang (lees: voor te veel epochs) getraind wordt voor de beschikbare hoeveelheid data. Het is dus belangrijk dat het model traint voor de juiste hoeveelheid epochs. Te weinig epochs zorgt ervoor dat het model niet zijn maximale performance bereikt, te veel epochs zorgt voor *overfitting*. Gelukkig bestaat er een oplossing om dit grotendeels te automatiseren: *early stopping*. (Ying, 2019)

Early stopping is een regularisatietechniek die kan gebruikt worden bij elke iteratieve trainingstechniek (zoals *gradient descent*). Het zorgt ervoor dat de training automatisch gestopt wordt als de performantie van het model op een validatieset

verslechterd. Daarnaast worden ook geen kostbare resources of tijd verspild. (Prechelt, 1998) In tegenstelling tot Keras en TensorFlow, bevat het PyTorch-framework geen geïntegreerde callback voor early stopping. Gelukkig is dit een relatief eenvoudige klasse om zelf te implementeren.

Voor een simpele `EarlyStopping`-klasse zijn er slechts enkele parameters nodig:

- `patience`: het aantal epochs waar geen verbetering plaatsvindt voordat het trainingsproces gestopt wordt.
- `delta`: de minimumwaarde die als verbetering wordt gezien
- `mode`: doel van het algoritme. Is het de bedoeling dat de metriek geminimaliseerd of gemaximaliseerd wordt?

Het eigenlijke `EarlyStopping` is verassend eenvoudig. Elke epoch wordt de `__call__`-methode aangeroepen. Afhankelijk van de `mode` wordt een score berekend. Deze score is gelijk aan de metriek die aan de methode meegegeven wordt als het doel is om deze te maximaliseren. Als deze metriek geminimaliseerd moet worden, is de score de tegengestelde van de metriek. Als er nog geen beste score is, wordt de huidige score als beste opgeslagen en wordt de huidige staat van het model als beste staat opgeslagen.

Als de score echter kleiner is dan dan de som van de beste score en de `delta`, wordt dit gezien als verslechtering. Een counter wordt verhoogd met 1 en als deze counter groter wordt dan de `patience`, dan wordt de training gestopt. Als geen van deze twee voorwaarden geldt (er is dus wel degelijk een significante verbetering), dan wordt de beste score aangepaste aan de huidige score, de huidige staat van het model als beste staat opgeslagen en de counter gereset. Eens het trainingsproces gestopt is, wordt de beste staat van het model terug ingeladen.

### Mixed precision training

Bij het trainen van objectdetectiemodellen is rekenkracht vaak een beperkende factor vanwege de hoge complexiteit van de netwerken en de omvangrijke input-data. De modellen die in dit onderzoek gebruikt zullen worden vereisen aanzienlijke GPU-geheugenruimte en rekestijd, vooral bij gebruik van leertechnieken zoals `SSL` en `Self-SL`, doordat ze deels of volledig zonder gelabelde data werken en vertrouwen op complexe pretext-taken of iteratieve labeling. In dit kader biedt *mixed precision training* een belangrijke optimalisatiemogelijkheid. Het zorgt namelijk voor een verminderd geheugengebruik en een sneller trainingsproces, zonder noemenswaardig verlies aan modelnauwkeurigheid. Dit maakt het mogelijk om efficiënter te trainen op dezelfde hardware, grotere batchgroottes te gebruiken en

sneller te experimenteren, wat van groot belang is tijdens dit iteratief ontwikkelingsproces.

Mixed precision training is een techniek die gebruik maakt van zowel 16-bit (half precision) als 32-bit (single precision) floating point getallen tijdens het trainen van neurale netwerken. In plaats van alle berekeningen in 32-bit uit te voeren – wat standaard is in de meeste frameworks – maakt mixed precision gebruik van 16-bit waar mogelijk, zonder daarbij de modelnauwkeurigheid significant te verliezen. Deze aanpak leidt tot versnelde trainingstijden en lager gebruik van GPU-geheugen.

In PyTorch wordt mixed precision training ondersteund via het **Automatic Mixed Precision (AMP)**-systeem, dat beschikbaar is in de `torch.amp` module. Met AMP kan mixed precision eenvoudig in de trainingscode geïntegreerd worden, zonder handmatig de precisie van elke operatie te moeten beheren. Het belangrijkste onderdeel van AMP is de autocast *context manager*, die automatisch beslist welke operaties in 16-bit uitgevoerd kunnen worden en welke in 32-bit moeten blijven. Daarnaast zorgt de *GradScaler utility* ervoor dat de gradiënten geschaald worden tijdens **backpropagation** om het risico op **buffer underflow** bij zeer kleine 16-bit waarden te beperken. Hierdoor blijft het trainingsproces stabiel.

Het gebruik van AMP is bijzonder waardevol in resource-intensieve taken zoals in dit onderzoek. Deze **SSL** en **Self-SL** objectdetectietaken vragen veel geheugen en rekenkracht. Door mixed precision toe te passen, kunnen grotere batch sizes worden gebruikt en snellere iteraties worden bereikt op dezelfde hardware. Bovendien ondersteunt de meeste moderne hardware van NVIDIA mixed precision training volledig, wat leidt tot aanzienlijke prestatieverbeteringen zonder kwaliteitsverlies.

## 4.2. Baseline: supervised Faster R-CNN

Allereerst is er een baseline-model nodig om de performance van de andere modellen mee te kunnen vergelijken. Dit model zal op de klassieke manier – namelijk gesuperviseerd – getraind worden. Dit houdt in dat het model tijdens het trainingsproces voor elke input een output heeft. Het model kan dus voor elke sample het verband leren tussen de features en de labels. De gesuperviseerde baseline zal daarnaast ook gebruikt worden als backbone voor de andere modellen (cf. infra).

In dit onderzoek werden al verschillende populaire supervised objectdetectiemodellen besproken. Hoewel deze allemaal geschikt zijn om als baseline te dienen, zal maar één van deze modellen geïmplementeerd worden. Dit om de simpele reden dat dit slechts een proof of concept is en dat de scope anders te ver zou worden uitgebreid. Het model dat geïmplementeerd zal worden is Faster **R-CNN**. De argumenten hiervoor zijn terug te vinden in sectie 3.3 van de methodologie.

Er zullen uiteindelijk verschillende varianten van hetzelfde model gebruikt worden in dit onderzoek. De eerste variant is een Faster R-CNN model dat getraind is op de volledige trainingsset (7600 samples). Deze variant zal gebruikt worden als baseline om de performantie van de andere modellen (SSL en Self-SL) mee te vergelijken. De andere varianten zijn qua architectuur volledig gelijk aan de eerste. Het verschil is dat deze getraind zullen worden op subsets van de trainingsset. Hier wordt dieper op ingegaan in sectie 3.2 van de methodologie. Voor de backbones van de andere leertechnieken zal telkens 5% en 10% van de gesuperviseerde data gebruikt worden. Deze modellen zullen daarna gebruikt worden voor de SSL- en Self-SL-modellen. Dit onderzoek zal voor het gesuperviseerde model het experiment in het onderzoek van Xie e.a. (2022) zo goed mogelijk proberen nabootsen. Aangezien dezelfde dataset gebruikt wordt in dit onderzoek, zullen de resultaten grotendeels hetzelfde zijn.

Specifiek zal een Faster R-CNN-model met een gepretrainde ResNet-18 backbone gebruikt worden. Dit model wordt in het onderzoek van Xie e.a. (2022) ook gebruikt en heeft een goede performance op de dataset. Zowel het model als de op ImageNet gepretrainde backbone zijn beschikbaar in de TorchVision bibliotheek. Om ze te gebruiken moeten ze enkel geïmporteerd worden, wat onnodig werk en tijd uitspaart ten opzichte van ze zelf te implementeren en volledig opnieuw te trainen. Er wordt gebruikgemaakt van een Adam-optimizer met een learning rate die lineair opgewarmd wordt voor de eerste 500 trainingsstappen. Hierbij wordt de learning rate geleidelijk verhoogd van  $1 \times 10^{-4}$  tot  $5 \times 10^{-4}$ . Daarnaast wordt een extra optimalisatie toegepast die ook in de paper voorkomt. Op epoch 8 en epoch 11 wordt de learning rate verminderd met 0.1. In totaal zal er 12 epochs gefinetuned worden. Hierbij wordt geen gebruik gemaakt van de EarlyStopping-callback om de resultaten van de paper zo goed mogelijk na te bootsen.

Tijdens het onderzoek werd er ook een poging gedaan om een geoptimaliseerde versie te maken van het trainingsscript. Dit script voerde enkele optimalisaties door die mogelijks de performantie van dit model konden verbeteren. Één van deze optimalisaties was het aanpassen van het vast aantal trainingsstappen waar de lineaire warm-up werd toegepast naar een variabel aantal stappen gebaseerd op een percentage van de hoeveelheid trainingsdata en het aantal epochs.

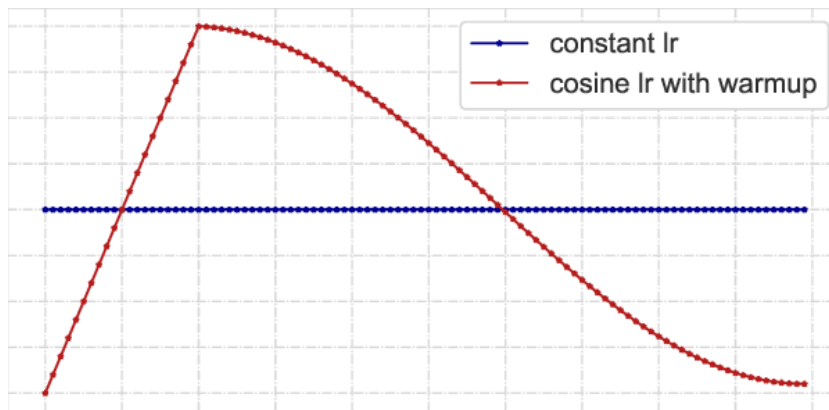
---

```
1 total_steps = len(train_loader) * epochs
2 warmup_iters = int(total_steps * warmup_pct)
3 warmup_iters = max(10, min(warmup_iters, 500))
```

---

**Codefragment 4.1:** Variabele warm-up logica voor het geoptimaliseerd trainingsscript van Faster R-CNN, gebaseerd op een percentage van de hoeveelheid trainingsdata en het aantal epochs.

Daarnaast wordt gebruik gemaakt van een AdamW-optimizer. Dit is betere versie van Adam. Hierbij wordt de *weight decay* ontkoppeld van de update-stap van de gradiënt. AdamW past de *weight decay* namelijk rechtstreeks tijdens de update van de parameters toe. Dit zorgt voor een betere generalisatie en dus een robuuster model. (Zhou e.a., 2024) De *learning rate* wordt overigens ook opgewarmd van  $1 \times 10^{-5}$  tot  $5 \times 10^{-4}$ . De *weight decay* bedraagt  $1 \times 10^{-4}$ . Ook wordt gebruik gemaakt van een CosineAnnealingLR-scheduler. Deze scheduler begint met een hoge *learning rate* en laat deze snel afnemen naar een lage *learning rate*. (Loshchilov & Hutter, 2016). Figuur 4.1 toont hoe de *learning rate* evolueert als ook warm-up wordt gebruikt – zoals in dit script het geval is. In tegenstelling tot het gewone Faster R-CNN-model, wordt hierbij wel gebruik gemaakt van EarlyStopping. De mAP op een validatieset wordt gemaximaliseerd met een patience van 10 epochs en een delta van 0.001.



**Figuur 4.1:** . Vergelijking tussen constante *learning rate* en *learning rate* aangepast door een Cosine Annealing LR-scheduler met warm-up. (Lu e.a., 2019)

Dit geoptimaliseerd model werd echter niet gebruikt in het uiteindelijke onderzoek om de resultaten zo goed mogelijk met de originele paper van Xie e.a. (2022) te kunnen vergelijken enerzijds en omdat de performantie van het model ondermaats was anderzijds.

### 4.3. Semi-supervised model (FixMatch)

Vervolgens wordt het SSL-model geïmplementeerd. Dit is een FixMatch-model. Details over de keuze van de architectuur kunnen teruggevonden in sectie 3.3 van de methodologie. Aangezien dit een semi-supervised architectuur is, zal het model getraind worden op een hybride manier. Specifiek zullen er twee modellen getraind worden. Enerzijds met een verdeling van 10% gelabeld / 90% ongelabeld, anderzijds met een verdeling van 5% gelabeld / 95% ongelabeld. Deze twee modellen laten het toe de leertechniek effectief te evalueren en de verbetering in performantie beter te begrijpen.

Als optimizer wordt **Stochastic Gradient Descent (SGD)** gebruikt, met aparte **learning rate** voor de backbone (0.0005) en de head (0.005), een momentum van 0.9 en een weight decay van 0.0005. Er wordt gebruik gemaakt van een **CosineAnnealingLR**-scheduler, mixed precision learning met PyTorch **AMP** en **EarlyStopping**.

Bij elke trainingsstap maakt het model gebruik van een batch van een inputafbeelding en de labelinformatie van de gesuperviseerde dataset en een (andere) inputafbeelding van de ongesuperviseerde dataset. Aangezien de ene dataset significant kleiner is dan de andere, zullen de gelabelde afbeeldingen dus hergebruikt worden om de zoveel trainingsstappen. Eerst en vooral wordt het model getraind op de gesuperviseerde batch. Vervolgens worden er twee verschillende augmentaties gemaakt van de ongesuperviseerde afbeelding: een zwakke en een sterke.

Daarna worden de pseudo-labels gegenereerd. Dit gebeurt door inferentie uit te voeren (door hetzelfde model) op de batch van de zwakke augmentaties van de ongesuperviseerde afbeeldingen en dus de **bounding boxes** en classificatielabels te voorspellen. Vervolgens worden de voorspellingen gefilterd om te voorkomen dat het model zou leren van foute voorspellingen. Deze filterstap steunt op twee verschillende condities. Eerst en vooral moet de **confidence score** groter zijn dan de huidige threshold. Deze threshold wordt berekend op basis van enkele parameters, waaronder de best behaalde **mAP** op de validatieset tot nu toe. De volledige formule voor de threshold is:

$$threshold_{current} = threshold_{min} + (threshold_{max} - threshold_{min}) \times \sqrt{mAP_{best}}$$

Hierbij zijn  $threshold_{min}$  en  $threshold_{max}$  parameters die aangepast kunnen worden. Tijdens de experimenten in dit onderzoek werden deze respectievelijk op 0.5 en 0.95 ingesteld. De vierkantswortel van de beste **mAP** op de validatieset wordt gebruikt om de groei in het begin te beperken en het model dus gestaag te laten verbeteren.

De tweede conditie van de filter controleert dat de oppervlakten van de voorspelde **bounding boxes** groter is dan  $8 \times 8 = 64$  pixels. Zo worden de **bounding boxes** met een onmogelijk klein oppervlak – en dus fout zijn – niet gebruikt om het model verder te trainen.

Eens de pseudo-gesuperviseerde data gefilterd is, kan ze gebruikt worden om het model verder te trainen. Echter wordt het model niet rechtstreeks getraind op deze data, maar op de sterke augmentatie van de overeenkomstige afbeeldingen. Dit maakt het model enorm robuust tegen verschillende augmentaties. Het 10%-model is voor 22 epochs (3:43:15 uur) gefinetuned en het 5%-model voor 24 epochs (4:15:42 minuten).

## 4.4. Self-supervised model (BYOL)

Uiteindelijk wordt het Self-SL-model geïmplementeerd aan de hand van het BYOL-framework. Details over de keuze voor dit model en de gebruikte architectuur zijn te vinden in sectie 3.3 van de methodologie. De implementatie verloopt in twee afzonderlijke fases: een pre-trainingsfase en een downstream-trainingsfase. De implementatie is grotendeels gebaseerd op die van Adaloglou (2022).

### 4.4.1. Pre-trainingsfase

De eerste fase focust op het trainen van een BYOL-model op ongesuperviseerde data. In deze fase is het doel om een encoder te trainen die sterke representaties leert extraheren uit ruwe beelden, zonder gebruik te maken van gelabelde data. Voor deze encoder wordt een ResNet-18 gebruikt als backbone, die vooraf werd geïnitieerd met ImageNet-gewichten. Het BYOL-mechanisme werkt met een online en een target netwerk, waarbij beide deze backbone gebruiken. De output van deze fase is een goed getraind representatiemodel dat gebruikt zal worden in de volgende stap.

Tijdens de training wordt een Adam-optimizer met een *learning rate* van  $3 \times 10^{-4}$  gebruikt. Daarnaast wordt tijdens deze fase gebruik gemaakt van mixed precision training met PyTorch AMP. Ook wordt EarlyStopping gebruikt om *overfitting* te voorkomen. Aangezien dit een pre-trainingsfase is, gebeurt de check op de *loss-functie*. De encoder werd 22 epochs getraind voor het EarlyStopping-mechanisme de training stopte. Dit duurde in totaal 1:33:08 uur. Uiteindelijk werd enkel de encoder van het online netwerk opgeslagen. Dit is dus de originele – en nu deels gefinetuneerde – ResNet-18-architectuur.

### 4.4.2. Downstream objectdetectietaak

In de tweede fase wordt het getrainde BYOL-model geïntegreerd als backbone in een Faster R-CNN model voor objectdetectie. In deze fase wordt het model getraind op gelabelde data, met als doel de representaties van de BYOL-backbone optimaal te benutten voor een supervisietaak.

Bij het trainen van dit downstream-model worden de volgende trainingsstrategieën toegepast om de performantie en stabiliteit te verbeteren:

- Bevriezing van lagen: De eerste twee lagen van de backbone (layer1 en layer2) worden gedurende de eerste drie epochs bevroren om de eerder geleerde representaties te behouden en verstoring tijdens het finetunen te voorkomen.
- Warm-up strategie: De leersnelheid wordt gradueel verhoogd van een initiële waarde van  $1 \times 10^{-5}$  naar een basiswaarde van 0.001.
- Optimizer: Er wordt gebruik gemaakt van SGD met een momentum van 0.9 en een weight decay van  $1 \times 10^{-4}$ .
- Scheduler: Er wordt gebruik gemaakt van een CosineAnnealingLR-scheduler.
- EarlyStopping: Om overfitting te vermijden, wordt EarlyStopping gebruikt tijdens de training.

De training van het objectdetectiemodel gebeurt slechts op een subset van de gelabelde dataset. Er worden twee scenario's onderzocht: een verhouding van 10% gelabelde data en 90% ongelabelde data, en een tweede, extremere configuratie met 5% gelabeld en 95% ongelabeld. Dit maakt het mogelijk om de effectiviteit van de BYOL-pre-training te evalueren in situaties met weinig gelabelde voorbeelden. Het 10%-model is voor 75 epochs (1:18:08 uur) gefinetuned en het 5%-model voor 45 epochs (45:40 minuten).

Samenvattend bestaat deze implementatie uit een pre-trainingsfase waarin een krachtige encoder wordt getraind via BYOL op ongesuperviseerde data, gevolgd door een finetuningfase waarin deze encoder geïntegreerd wordt in een Faster R-CNN detectiemodel. Dankzij het gebruik van een pre-trained backbone, gecontroleerde fine-tuning, en strategieën zoals warm-up, bevroren lagen en EarlyStopping, wordt verwacht de prestaties van het eindmodel te optimaliseren onder beperkte supervisie.

# 5

## Resultaten en evaluatie

In dit hoofdstuk worden de resultaten van de verschillende getrainde modellen met elkaar vergeleken. De focus ligt daarbij op drie grote categorieën van trainingsstrategieën: supervised learning, **SSL** (hier vertegenwoordigd door FixMatch) en **Self-SL** (hier via **BYOL**). Door deze methodes systematisch te vergelijken, kan worden nagegaan hoe goed elk type model presteert bij een gelimiteerde hoeveelheid gelabelde data, en welke strategie het meest efficiënt is in het benutten van die data.

### 5.1. Resultatenvergelijking: supervised vs SSL vs Self-SL

Supervised learning vormt de klassieke aanpak waarbij modellen uitsluitend trainen op volledig gelabelde datasets. In dit onderzoek werden modellen getraind met verschillende percentages gelabelde data: 1%, 5%, 10%, 50% en 100%. Deze variatie biedt inzicht in hoe sterk de performantie van een model afhankelijk is van de hoeveelheid supervisie, en dient als referentiepunt voor de andere methodes.

De **SSL** aanpak via FixMatch combineert een kleine hoeveelheid gelabelde data met een grotere hoeveelheid ongelabelde data, waarbij het model zelf pseudo-labels genereert op basis van zijn voorspellingen. Deze pseudo-labels worden vervolgens gebruikt om het model verder te trainen. FixMatch is specifiek ontworpen om sterke resultaten te behalen met een beperkte gelabelde subset, door slim gebruik te maken van augmentaties en consistentie tussen voorspellingen.

**Self-SL**, in dit geval via **BYOL**, leert representaties zonder enige labelinformatie, door middel van contrastieve technieken waarbij het model leert om verschillende weergaven van hetzelfde beeld te associëren. Deze representaties worden vervolgens gebruikt om een objectdetector te trainen met beperkte supervisie. **Self-SL** is vooral

interessant omwille van zijn label-onafhankelijkheid in de pretrainingsfase.

De vergelijking tussen deze methodes biedt niet enkel inzicht in de accuraatheid en generalisatiecapaciteit van de modellen, maar legt ook bloot hoe efficiënt elk type benadering omgaat met beperkte annotatie. Daarbij wordt zowel een kwantitatieve (via **mAP**) als kwalitatieve analyse uitgevoerd, om de sterktes en beperkingen van elke methode in de praktijk te evalueren.

## 5.2. Kwantitatieve analyse

De prestaties van de verschillende modellen en trainingsstrategieën zijn overzichtelijk samengevat in Tabel 5.1. Deze tabel toont de **mAP** scores voor Faster **R-CNN** bij verschillende percentages gelabelde data, alsook de resultaten van **SSL** FixMatch en **Self-SL** **BYOL** modellen getraind op respectievelijk 5% en 10% gelabelde data.

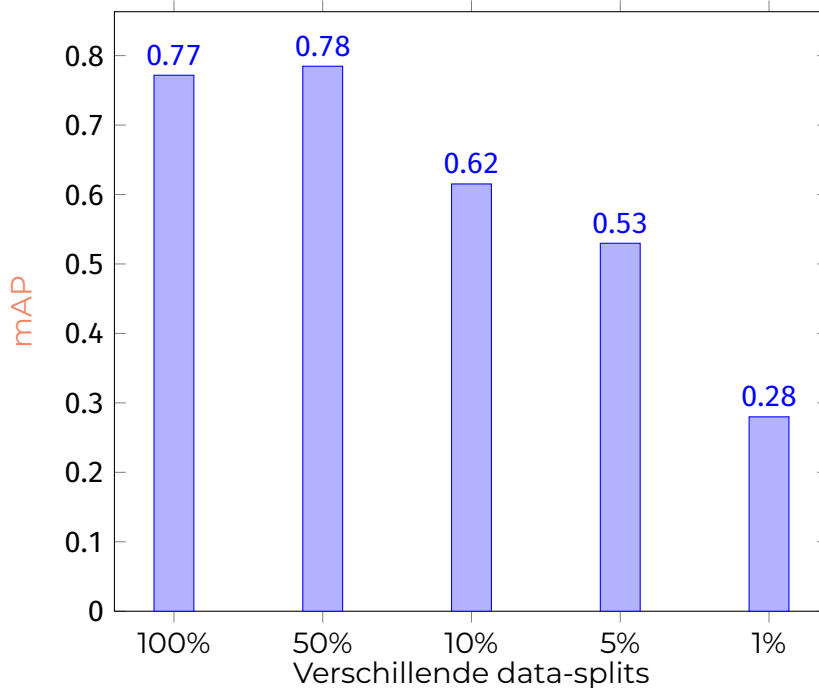
Model en Gelabeld Percentage	mAP
Faster R-CNN 1%	0.2799
Faster R-CNN 5%	0.5298
Faster R-CNN 5% + BYOL	0.6452
Faster R-CNN 10%	0.6152
Faster R-CNN 10% + BYOL	0.7230
Faster R-CNN 50%	0.7847
Faster R-CNN 100%	0.7717
FixMatch 5%	0.6649
FixMatch 10%	0.6828

**Tabel 5.1:** Overzicht van **mAP**-resultaten voor verschillende modellen en gelabelde datavolumes.

De evaluatie van de verschillende modellen en trainingsstrategieën laat duidelijke verschillen zien in prestaties gemeten aan de hand van de **mAP**. Het baseline Faster **R-CNN** model, getraind op de volledige dataset (100% gelabelde data), behaalt een **mAP** van 0.7717, wat een solide uitgangspunt vormt voor vergelijking met modellen die minder gelabelde data gebruiken of **SSL** en **Self-SL** technieken toepassen.

Wanneer de hoeveelheid gelabelde data wordt teruggebracht, daalt de prestaties van het supervised Faster **R-CNN** model significant. Bij 50% gelabelde data stijgt de **mAP** zelfs licht tot 0.7847, mogelijk door toeval of lichte verschillen in splitsing, maar bij verdere reductie naar 10%, 5% en 1% zakt de **mAP** respectievelijk naar 0.6152, 0.5298 en 0.2799. Dit bevestigt het belang van voldoende gelabelde data voor goede prestaties in traditionele supervised learning.

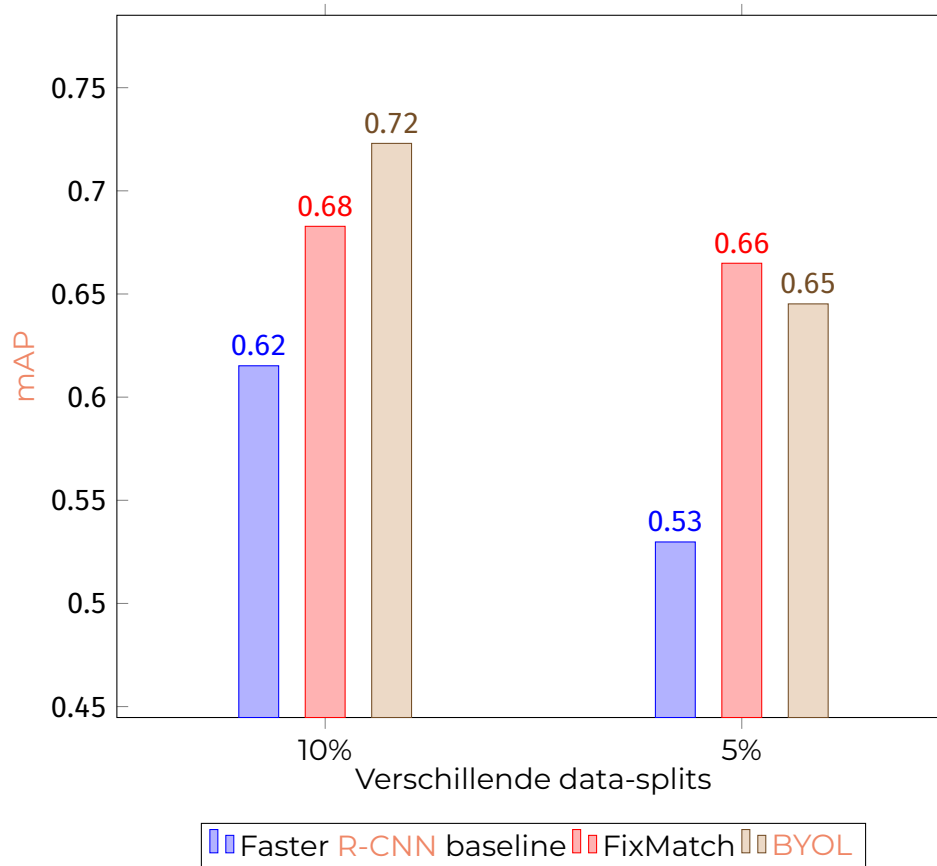
Figuur 5.1 visualiseert deze trend duidelijk. Hierin is zichtbaar dat de prestatie van het supervised model exponentieel afneemt naarmate de hoeveelheid gelabelde data wordt teruggeschoefd, wat de uitdaging benadrukt van voldoende annotations in sonardatasets.



**Figuur 5.1:** Overzicht van de mAP voor de supervised Faster R-CNN modellen getraind op de verschillende data-splits.

De toepassing van Self-SL pretraining via BYOL, gevolgd door fine-tuning met beperkte gelabelde data, laat een merkbare verbetering zien ten opzichte van het pure supervised model bij lage data-volumes. Figuur 5.2 toont deze vergelijking van mAP-scores. Zo verbetert de mAP bij 5% gelabelde data van 0.5298 (Faster R-CNN) naar 0.6452 (Faster R-CNN + BYOL), en bij 10% gelabelde data van 0.6152 naar 0.7230. Dit suggereert dat Self-SL pretraining effectief representaties leert die waardevol zijn voor downstream objectdetectie, waardoor het model minder afhankelijk wordt van grote hoeveelheden gelabelde data.

FixMatch, als SSL methode, toont eveneens verbetering boven de pure supervised baseline bij 5% en 10% gelabelde data, met mAP-waarden van respectievelijk 0.6649 en 0.6828. Hoewel FixMatch op 10% iets onder het BYOL-gefine-tuned model blijft, overtreft het de baseline aanzienlijk, wat de potentie van SSL in dit domein onderstreept.



**Figuur 5.2:** Overzicht van de mAP voor het supervised Faster R-CNN model, het SSL FixMatch model en het Self-SL BYOL model, getraind op 10% en 5% data-splits.

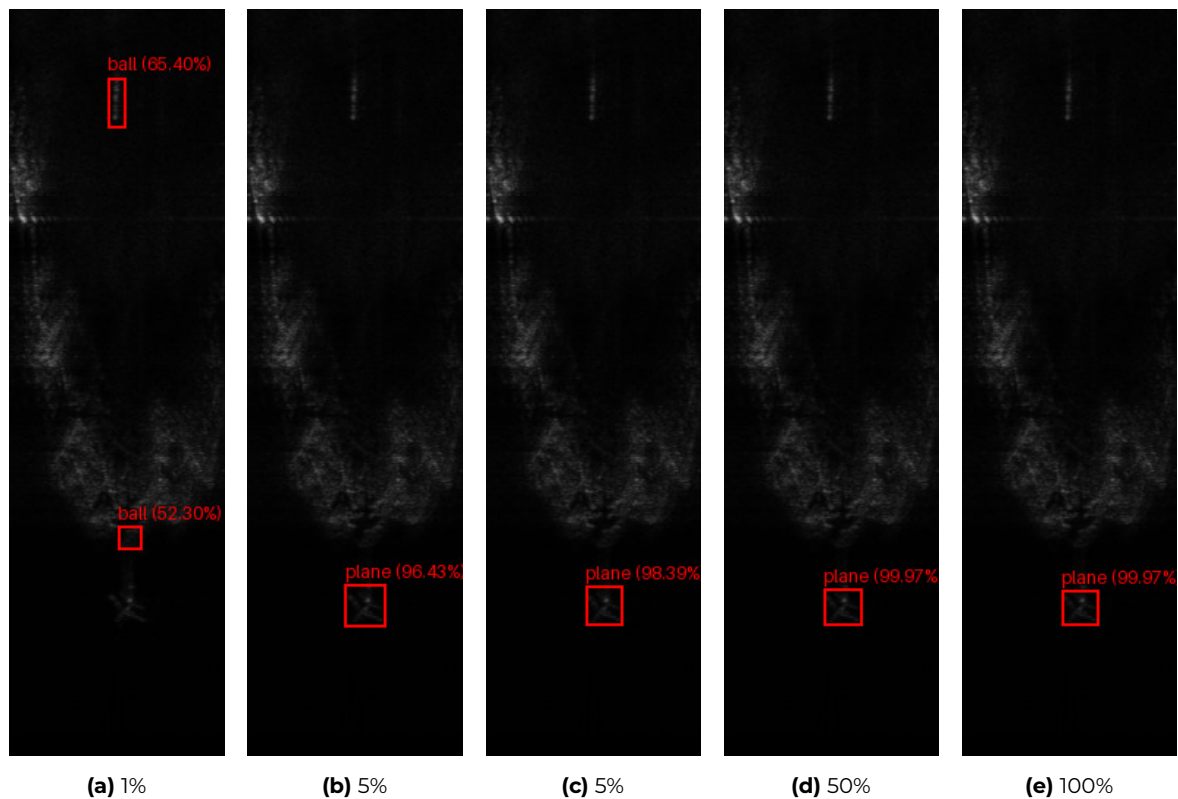
Deze resultaten impliceren dat de combinatie van representatielernen via Self-SL pretraining of SSL label-augmentatie het model minder afhankelijk maakt van grote hoeveelheden gelabelde data, wat cruciaal is voor kostbare domeinen als sonardata-analyse. De relatief kleine verschillen tussen FixMatch en BYOL kunnen wijzen op verschillende sterke kanten van SSL versus Self-SL methoden, afhankelijk van de dataset en de implementatie.

Daarnaast blijkt dat bij grotere hoeveelheden gelabelde data (50% en 100%) het pure supervised model nog steeds het beste scoort, hoewel de winst bij 100% ten opzichte van 50% beperkt is. Dit kan een indicatie zijn van een verzadigingseffect waarbij extra gelabelde data slechts marginale verbeteringen opleveren. Het SSL en Self-SL trainen zijn dan minder noodzakelijk, maar kunnen nog steeds nuttig zijn om trainingskosten en annotatielast te reduceren.

### 5.3. Kwalitatieve analyse

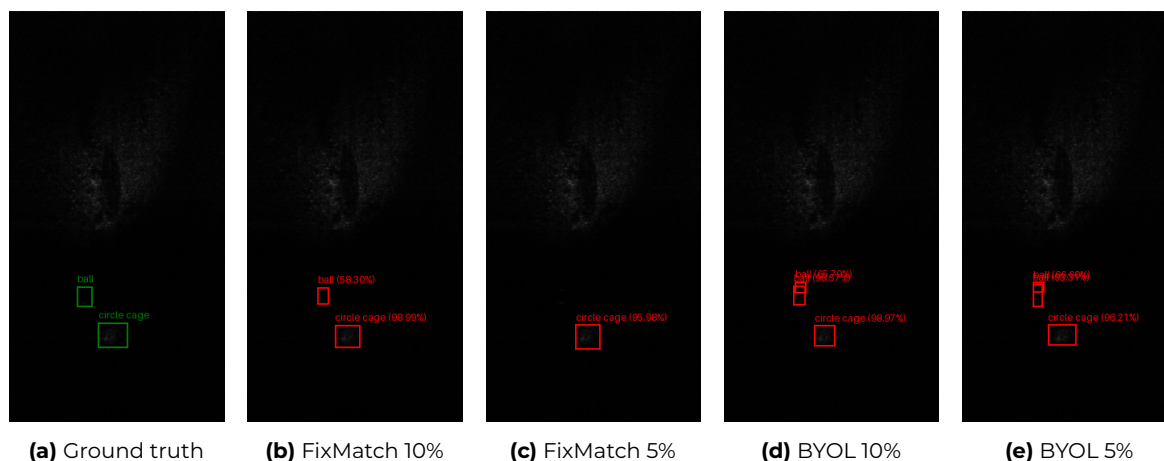
Op basis van de visuele inspectie van de modelvoorspellingen biedt de kwalitatieve analyse waardevolle inzichten in het gedrag en de betrouwbaarheid van de verschillende detectiemodellen. De analyse is gebaseerd op twee figuren: één die

de prestaties van de supervised modellen toont, en een tweede die de prestaties van de onderzochte SSL en Self-SL modellen (FixMatch en BYOL) vergelijkt met de ground truth.



**Figuur 5.3:** Voorspellingen op afbeelding 00251 van de UATD\_Test\_1-dataset door Faster R-CNN getraind op verschillende subsets.

In de figuur met alle supervised modellen valt op dat de prestaties van de meeste modellen relatief sterk zijn, zelfs bij beperkte hoeveelheden gelabelde data. De detectie-outputs van de modellen getraind op 5%, 10%, 50% en 100% gelabelde data tonen telkens correcte identificaties van de objecten, met nauwkeurig geplaatste **bounding boxes** en hoge **confidence scores**. Enkel het model getraind met slechts 1% gelabelde data wijkt opvallend af. Dit model genereert foutieve voorspellingen, herkent objecten die niet aanwezig zijn, en doet dit bovendien met hoge **confidence scores**. Ook worden aanwezige objecten fout geclassificeerd. Deze bevinding bevestigt dat bij een te beperkte hoeveelheid gelabelde data het supervised model niet in staat is robuuste representaties te leren, wat leidt tot onbetrouwbare detectie.



**Figuur 5.4:** Voorspellingen op afbeelding 00001 van de UATD\_Test\_1-dataset door FixMatch en Faster R-CNN + BYOL getraind op verschillende subsets.

In de tweede figuur, waarin de voorspellingen van FixMatch en BYOL vergeleken worden met de ground truth, komen enkele opvallende verschillen naar voren. Het FixMatch-model getraind met 10% gelabelde data voorspelt alle objecten correct, maar doet dit met relatief lage *confidence scores* voor bepaalde objecten. Dit suggereert dat het model wel degelijk leert om relevante objectkenmerken te herkennen, maar minder zeker is van zijn voorspellingen – mogelijk door de beperkte kwaliteit van de pseudo-labels tijdens het trainingsproces. Bij het FixMatch-model getraind met slechts 5% gelabelde data ontbreekt één object volledig in de voorspellingen. Dit wijst op een verhoogde fragiliteit van het model bij nog beperktere labeling, wat impliceert dat de hoeveelheid gelabelde data bij FixMatch sterk correleert met de volledigheid van de voorspellingen.

De BYOL-gebaseerde modellen vertonen een ander gedragspatroon. Zowel bij 5% als bij 10% gelabelde data worden alle objecten correct herkend en geclassificeerd. Dit bevestigt dat de representaties die via BYOL geleerd worden, effectief zijn in het ondersteunen van objectdetectie bij gelimiteerde data. Echter, er worden opvallend veel *bounding boxes* over elkaar heen voorspeld. Dit fenomeen duidt mogelijk op een tekort aan post-processing, met name het ontbreken of incorrect toepassen van NMS. Hoewel de onderliggende herkenning goed is, leidt dit gedrag tot redundante output die de interpretatie bemoeilijkt en de precisie negatief kan beïnvloeden.



**Figuur 5.5:** Correcte voorspellingen van Faster R-CNN 5% + BYOL (afbeelding 00008 van de UATD\_Test\_1-dataset)

De kwalitatieve analyse bevestigt dat het gebruik van SSL en Self-SL strategieën zinvol is bij gelimiteerde labeling. FixMatch lijkt gevoeliger voor labelhoeveelheid dan BYOL, terwijl BYOL robuuste detecties levert, maar extra optimalisatie vereist op het niveau van outputfiltering. De vergelijking met de supervised modellen toont aan dat vanaf 5% gelabelde data al vrij betrouwbare prestaties gehaald worden, maar dat de meer geavanceerde pretrainingstrategieën een duidelijke meerwaarde kunnen bieden voor nauwkeurigheid en consistentie – mits ze correct worden afgestemd.

# 6

## Conclusie

### 6.1. Samenvatting van bevindingen

Op basis van de uitgevoerde experimenten kan geconcludeerd worden dat zowel **SSL** als **Self-SL** effectieve strategieën zijn om de afhankelijkheid van gelabelde data bij objectdetectie in sonarbeelden te reduceren. Het volledig gesuperviseerde baseline-model, Faster **R-CNN**, behaalde een hoge nauwkeurigheid wanneer het getraind werd op de volledige gelabelde dataset (100%), met een **mAP** van 0.7717. Bij afname van het gelabelde aandeel daalden de prestaties echter significant, met **mAP**-scores van respectievelijk 0.7847 (50%)<sup>1</sup>, 0.6152 (10%), 0.5298 (5%) en 0.2799 (1%). Deze trend onderstreept de kwetsbaarheid van conventionele supervised modellen in situaties met beperkte annotaties.

De toepassing van **SSL** met FixMatch bood in dit opzicht een duidelijke verbetering. Wanneer slechts 5% of 10% van de data gelabeld was, behaalde FixMatch **mAP**-scores van 0.6649 en 0.6828. Deze scores liggen beduidend hoger dan die van het supervised baseline-model bij dezelfde splits, wat aantoont dat het model in staat is om op effectieve wijze gebruik te maken van de resterende ongelabelde data via pseudo-labeling en consistency regularisatie. Hoewel FixMatch niet de absolute topresultaten behaalde, vormt het een krachtige techniek voor situaties waarin annotaties beperkt beschikbaar zijn, zonder dat pretraining noodzakelijk is.

De best presterende alternatieve aanpak bleek echter de **Self-SL** strategie met **BYOL**. Door een **BYOL**-model te pretrainen op ongelabelde sonarbeelden en deze gewichten vervolgens te gebruiken als backbone voor een Faster **R-CNN** model, konden bij

<sup>1</sup>De hogere performantie van het 50%-model ten opzichte van het 100%-model kan verklaard worden door **overfitting**: het model dat op 100% van de data is getraind, heeft mogelijk meer ruis of irrelevante patronen in de volledige dataset geleerd. Bij 50% gelabelde data is de kans groter dat het model generaliseert naar meer robuuste kenmerken, wat resulteert in een iets hogere **mAP**.

5% en 10% gelabelde data **mAP**-scores van respectievelijk 0.6452 en 0.7230 bereikt worden. Met name de score van 0.7230 bij 10% gelabelde data benadert de prestaties van het volledig gesuperviseerde model, ondanks dat het met slechts een tiende van de annotaties getraind werd. Dit suggereert dat de representaties geleerd tijdens de **BYOL**-pretraining bijzonder effectief generaliseren naar de downstream detectietaak.

Samenvattend tonen de resultaten aan dat zowel FixMatch als **BYOL** waardevolle technieken zijn in data-schaarsere omgevingen. FixMatch biedt een efficiënte manier om ongelabelde data direct tijdens training te benutten, terwijl **BYOL** robuuste representaties leert die downstream training aanzienlijk versterken. De **Self-SL** aanpak met **BYOL** leverde de beste algehele prestaties in low-label settings, en vormt daarmee een veelbelovende richting voor verdere optimalisatie van objectdetectie in gespecialiseerde domeinen zoals sonarbeeldanalyse.

## 6.2. Reflectie op de methodologie

Bij reflectie op de gehanteerde methodologie kan worden vastgesteld dat de keuzes die aan de basis van dit onderzoek lagen over het algemeen sterk onderbouwd en passend waren binnen de onderzoeksvraag. De geselecteerde dataset bleek bijzonder geschikt: ze bood voldoende variatie en detail om de relevantie van het probleem van objectdetectie in sonarbeelden realistisch te benaderen. Bovendien sloot de datastructuur goed aan bij de gekozen leerstrategieën, wat toeliet om zowel supervisie als pretraining correct te benutten.

Ook de selectie van modellen – Faster **R-CNN**, FixMatch en **BYOL** – bleek achteraf gezien goed doordacht. Deze modellen zijn zowel conceptueel uitdagend als voldoende krachtig om degelijke experimentele resultaten op te leveren. Ze boden een goed evenwicht tussen theoretische diepgang en praktische toepasbaarheid. Tegelijkertijd moet worden erkend dat de implementatie van vooral de **SSL** en **Self-SL** modellen niet zonder moeilijkheden verliep. Het FixMatch-model, hoewel goed gedocumenteerd in literatuur, bleek in praktijk lastig op punt te stellen: het verkrijgen van stabiele en competitieve prestaties vereiste veel iteraties, hyperparameter-tuning en finetuning van augmentatiestrategieën. Het model bleek gevoelig voor kleine variaties in input en training, wat het reproduceerbaar werken bemoeilijkte.

De moeilijkheden bij **BYOL** waren nog fundamenteler in de vroege fase. Initieel bleek dat een fout in de modelconfiguratie ertoe leidde dat slechts een beperkt deel van de ResNet-architectuur effectief getraind werd. Deze fout beperkte de prestaties drastisch tot **mAP**-scores onder 0.15, waardoor diepgaand debugging-werk noodzakelijk was. Pas na het herstellen van deze fout kon het **BYOL**-model zijn potentieel laten zien. Daarnaast was het uitvoeren van optimalisaties niet evi-

dent: kleine aanpassingen aan architectuur of trainingsstrategie hadden vaak geen effect of zelfs negatieve impact. Dit toont aan dat dergelijke modellen fragiel kunnen zijn en dat hun succes sterk afhankelijk is van zorgvuldige afstemming van alle onderdelen van de trainingspipeline.

Verder speelde de computationele requirements een belangrijke rol. Door de (relatief) lange trainingstijden en de nood aan krachtige GPU-resources, moest het onderzoek in iteraties worden uitgevoerd en werd experimenteel werk vaak vertraagd. Sommige experimenten konden niet in parallel worden opgezet, wat tijdsdruk met zich meebracht. Deze beperkingen illustreren de praktische uitdagingen bij het toepassen van geavanceerde leermethodes in een onderzoeksomgeving die beperkt wordt qua resources.

Ondanks deze obstakels was de gekozen methodologie waardevol. De moeilijkheden brachten leerzame inzichten met zich mee over de robuustheid en gevoeligheid van moderne machine learning-technieken in domeinspecifieke contexten. Het experiment gaf een realistisch beeld van wat er komt kijken bij het inzetten van geavanceerde SSL en Self-SL modellen in een complexe toepassing zoals sonarbeeldanalyse.

### 6.3. Voorstellen voor verder onderzoek

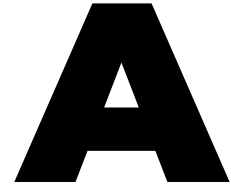
Hoewel dit onderzoek aantoont dat zowel SSL als Self-SL grote voordelen kunnen bieden bij objectdetectie op sonarbeelden, blijven er nog verschillende onderzoekspistes open die verdere verkenning verdienen. Een eerste richting is het vergelijken van alternatieve Self-SL pretrainingstrategieën. Hoewel BYOL in dit onderzoek werd gekozen vanwege zijn stabiliteit en prestaties zonder negatieve paren, zijn er andere competitieve methoden zoals SimCLR, MoCo en Self-Distillation with No Labels (DINO). Een vergelijking van deze methodes zou kunnen uitwijzen welke het meest geschikt is voor de specifieke kenmerken van sonarbeeldvorming, zoals lage resolutie, ruis en beperkte kleurinformatie. Daarnaast kan onderzocht worden hoe verschillende backbone-architecturen (bijvoorbeeld ResNet, Swin Transformer of ConvNeXt) zich verhouden qua compatibiliteit en prestaties binnen deze pretrainingsstrategieën.

Ook op het vlak van SSL zijn er waardevolle alternatieven die verder onderzoek nodig hebben. FixMatch toonde in dit onderzoek al sterke resultaten, maar recentere methoden zoals FlexMatch, SoftMatch of Unsupervised Data Augmentation (UDA) hebben mogelijk nog betere prestaties, zeker in combinatie met meer geavanceerde of domeinspecifieke augmentatietechnieken. Denk hierbij aan augmentaties die beter aansluiten bij sonarcontexten, zoals het toevoegen van realistische ruis, reflectievariaties of vervormingen die voortkomen uit onderwaterpropagatie.

Een bijzonder interessante piste is de combinatie van **Self-SL** pretraining en **SSL** fine-tuning. Door eerst representaties te leren via een **Self-SL** benadering zoals **BYOL** en deze vervolgens verder te verfijnen met een methode als FixMatch, kunnen de voordelen van beide strategieën worden gecombineerd. Experimenteel onderzoek moet uitwijzen of deze hybride aanpak leidt tot hogere nauwkeurigheid en robuustere detectieprestaties in situaties met een minimale hoeveelheid gelabelde data.

Verder kan onderzoek naar transfer learning binnen het sonardomein waardevolle inzichten opleveren. Modellen die getraind zijn op grootschalige ruwe sonardata uit verschillende toepassingen – bijvoorbeeld onderwaternavigatie, structuurinspectie of archeologisch onderzoek – kunnen geëvalueerd worden op hun vermogen om te generaliseren naar nieuwe omgevingen of sensorconfiguraties. Dit zou toelaten om universele sonarbackbones te ontwikkelen die met minimale aanpassing bruikbaar zijn in diverse domeinen.

Ten slotte is er nood aan onderzoek dat zich richt op de interactie tussen modelprestatie en labelselectie. Actief leren, waarbij het model zelf aangeeft welke samples het meest informatief zijn om te labelen, biedt hier een kansrijke uitbreiding. Door actief leren te combineren met **Self-SL** of **SSL** technieken, kan mogelijk een nog efficiëntere annotatiestrategie ontwikkeld worden. Daarnaast verdient ook foutenanalyse meer aandacht: een diepgaand inzicht in de soorten fouten die modellen maken, kan bijdragen aan interpretatie, vertrouwen en verdere optimalisatie van detectieprestaties in kritieke toepassingen zoals defensie of onderwaterrobotica.



## Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

### Samenvatting

Sinds de opkomst en popularisatie van AI-modellen is data steeds een cruciale resource geweest. Voor simpele modellen is de benodigde data vaak ook simpel van vorm en is er (relatief) weinig van nodig om een performant en goed werkend model te creëren. Echter stijgen de data-requirements voor grotere en complexere modellen exponentieel. De benodigde data om een objectdetectiemodel voor sonardata te trainen zorgt voor moeilijkheden: dit soort datasets zijn online niet off-the-shelf beschikbaar en zijn dus zeer tijdrovend en kostbaar om te maken. De hoofdvraag van dit onderzoek is daarom: Op welke manieren kan het gebruik van semi- of self-supervised learning het labelproces versnellen zonder significante verlies in nauwkeurigheid? Door verschillende technieken toe te passen, zal een pretraining-strategie ontwikkeld worden die gebruikmaakt van ongelabelde data om representaties aan te leren. Vervolgens zal onderzocht worden hoeveel gelabelde data nodig is om een goed presterend detectiemodel te trainen. Het doel is een methodologie te ontwikkelen die de afhankelijkheid van handmatig gelabelde data minimaliseert, terwijl de prestaties van het detectiemodel behouden blijven. De resultaten kunnen bijdragen aan efficiëntere workflows voor data-analyse in sonarbeeldvorming en andere domeinspecifieke contexten.

## A.1. Inleiding

Objectdetectie in sonar data speelt een cruciale rol in toepassingen zoals onderwaterverkenning, maritieme veiligheid en archeologisch onderzoek. Traditionele methoden voor objectdetectie vereisen echter grote hoeveelheden gelabelde data, wat een tijdrovend en duur proces is. Sonar afbeeldingen, gekenmerkt door ruis en unieke reflectiepatronen, vergen bovendien gespecialiseerde kennis voor annotatie. Dit maakt het labelen van datasets een grote uitdaging, vooral wanneer schaal en diversiteit van de data toenemen.

Semi- en self-supervised learning bieden veelbelovende alternatieven om dit probleem te overbruggen. Deze technieken maken gebruik van ongesuperviseerde data om representaties te leren en beperken de afhankelijkheid van gelabelde gegevens. Moderne self-supervised methoden hebben indrukwekkende resultaten laten zien in domeinen zoals computer vision, maar hun toepassing op domeinspecifieke datasets, zoals sonar, is nog relatief onbekend terrein.

Dit onderzoek richt zich op de vraag of semi- en self-supervised learning technieken effectief kunnen worden ingezet om het labelproces in sonar objectdetectie te versnellen, zonder dat dit ten koste gaat van de nauwkeurigheid van het model. Door technieken aan te passen aan de specifieke eigenschappen van sonar data, wordt onderzocht hoe representaties kunnen worden geleerd die bijdragen aan een efficiëntere en kosteneffectieve workflow.

Naast de vergelijking tussen semi-supervised / self-supervised en supervised learning, zullen verschillende details onderzocht worden. Dit gaat van hoeveelheid gelabelde data minimaal nodig om met semi-supervised learning vergelijkbare resultaten te behalen als volledig gesuperviseerde methoden tot pre-training-methoden uit self-supervised learning die het meest geschikt zijn voor sonar data. Daarnaast wordt onderzocht welke aanpassingen nodig zijn om bestaande semi- en self-supervised technieken effectief toe te passen op stacked GeoTIFF-sonar data. Uiteindelijk wordt uitgezocht welke evaluatiemethoden het beste de prestaties van semi- en self-supervised modellen in sonar objectdetectie kunnen meten.

## A.2. Literatuurstudie

Objectdetectie in domeinspecifieke contexten zoals sonar beeldvorming wordt vaak gehinderd door een gebrek aan gelabelde data. Traditioneel vereisen gesuperviseerde modellen grote hoeveelheden handmatig gelabelde gegevens om effectieve detectie en classificatie te leren. Semi-supervised en self-supervised learning bieden echter veelbelovende alternatieven door gebruik te maken van grote hoeveelheden ongesuperviseerde data om representaties te leren. Deze literatuurstu-

die bespreekt de huidige technieken en hun toepassing, met een specifieke focus op de unieke uitdagingen van sonardata.

### **A.2.1. Gesuperviseerde Objectdetectie**

Objectdetectie is een tak binnen het domein van computer vision dat gericht is op het identificeren en lokaliseren van objecten binnen beelddata (zoals foto's en video's). Dit wordt gebruikt in verschillende domeinen, zoals beveiligingssystemen (bv. om inbrekers te detecteren) of de medische wereld (bv. om tumoren op te sporen). Door de jaren heen is objectdetectie aanzienlijk geëvolueerd dankzij de vooruitgang in Deep Learning en de grote beschikbaarheid van datasets met beeldmateriaal. (He e.a., 2016)

Objectdetectie combineert twee belangrijke zaken in computer vision: objectlokalisatie en objectclassificatie. Objectlokalisatie bepaalt de positie van objecten, meestal in de vorm van bounding boxes (Tompson e.a., 2015), terwijl objectclassificatie bepaalt tot welke categorie een gedetecteerd object behoort. Samen geeft dit de mogelijkheid tot het herkennen van verschillende objecten op één foto.

Traditioneel worden gesuperviseerde methoden – zoals Faster R-CNN, YOLO en SSD – gebruikt voor objectdetectie. (Redmon e.a., 2016) Deze modellen presteren uitstekend bij voldoende gelabelde data, maar de annotatiekosten en tijdsinvestering vormen een grote belemmering, vooral bij complexe datasets zoals sonar. Sonardata vereist namelijk gespecialiseerde kennis voor het labelen, wat de annotatie nog uitdagender maakt. (Long e.a., 2015)

### **A.2.2. Semi-supervised learning**

Binnen het domein van machine learning bestaan er verschillende technieken om een model te trainen. Meestal wordt er gesproken van twee grote stromingen: supervised learning en unsupervised learning. Bij supervised learning wordt er gebruik gemaakt van een dataset en een uitkomst (hetgeen het model uiteindelijk moet kunnen voorspellen). Dit kan een label zijn of een bepaalde numerieke waarde. Belangrijk is dat zowel de input als de gewenste output gegeven zijn. Het model leert dus het verband tussen de twee. Bij unsupervised learning zijn er geen verwachte outputs. De volledige dataset wordt door het model gebruikt om patronen in te herkennen. Unsupervised learning wordt daarom ook meestal gebruikt om verkennende data-analyse uit te voeren. Echter hebben beide methoden enkele nadelen. Bij supervised learning is het traag en duur om alle data op een correcte manier te labelen. Unsupervised learning heeft dit probleem niet, maar heeft een beperkt aantal toepassingen en is minder accuraat. Een alternatief is semi-supervised learning, wat een compromis tussen zowel supervised als unsupervised learning is. (C A Padmanabha Reddy e.a., 2018)

Semi-supervised learning maakt gebruik van zowel gelabelde als ongelabelde data. Het algoritme traint eerst een basismodel op de beperkte hoeveelheid gelabelde data, waarna dit gebruikt wordt om de grotere ongelabelde dataset van labels te voorzien. Deze techniek wordt ook wel Pseudo-labeling genoemd. (D.-H. Lee, 2013)

Een andere populaire techniek binnen semi-supervised learning is Consistency Regularization. Deze techniek zorgt ervoor dat het model consistente voorspellingen aanleert door gebruik te maken van data-augmentatie. De inputdata wordt hierbij licht aangepast of verstoord. Het uitgangspunt is dat een goed model robuust moet zijn tegen kleine veranderingen in de input. (Fan e.a., 2022). Deze twee populaire technieken kunnen ook gebruikt worden om een nog beter model te maken zoals in het FixMatch-algoritme van Sohn e.a. (2020).

### **A.2.3. Self-supervised learning**

Anders dan bij semi-supervised learning is er bij self-supervised learning of SSL helemaal geen nood aan labels. Deze creëert het algoritme namelijk zelf tijdens een pre-training fase. Semi-supervised learning behoort echter niet helemaal tot het domein van unsupervised learning, hoewel het gebruik maakt van verschillende groeperings- en clusteringmethoden. Het uiteindelijke model maakt namelijk gebruik van – door de pre-training – gelabelde data. Deze techniek zorgt ervoor dat er veel complexere modellen getraind kunnen worden zonder een gigantische hoeveelheid aan data. Enkele nadelen zijn wel dat de techniek een grote hoeveelheid computerkracht nodig heeft en een lagere accuratie heeft dan supervised learning. (Gui e.a., 2024)

Ondanks deze nadelen blijken verschillende populaire technieken succesvol binnen het domein van computer vision.

- SimCLR: deze afkorting staat voor Simple Framework for Contrastive Learning of Visual Representations. Deze techniek leert door gelijkenissen en verschillen tussen gegevensvoorbeelden te vergelijken, gebruikmakend van augmentaties zoals rotatie en kleurverandering. Het model wordt getraind om representaties van dezelfde input dichter bij elkaar te brengen (positieve paren) en die van verschillende inputs verder uit elkaar te houden (negatieve paren). (T. Chen e.a., 2020)
- BYOL: deze afkorting staat voor Bootstrap Your Own Latent. Deze techniek gebruikt twee netwerken: een online netwerk en een target netwerk. Het online netwerk leert een representatie van de ene augmentatie. Hierna genereert het target netwerk een representatie van de andere augmentatie. Uiteindelijk wordt het model getraind om de representatie van het online netwerk zo

dicht mogelijk te laten lijken op die van het target netwerk. Merk op dat BYOL geen negatieve paren nodig heeft, wat leidt tot eenvoudigere training en vaak betere resultaten. (Grill e.a., 2020)

Zowel SimCLR als BYOL hebben voordelen en nadelen. Toch zijn ze allebei uitermate geschikt voor de use-case in dit onderzoek. Hieronder worden enkele verschillen besproken:

- Contrastive learning: SimCLR gebruikt contrastieve loss met zowel positieve als negatieve paren, terwijl BYOL gebruikt enkel positieve paren gebruikt.
- Efficiëntie: BYOL heeft minder geheugen en computationele middelen nodig omdat er geen grote batch negatieve voorbeelden nodig zijn.
- Complexiteit: SimCLR is eenvoudiger qua architectuur terwijl BYOL een dubbele netwerkstructuur nodig heeft.

Beide methoden hebben bewezen krachtig te zijn in self-supervised learning en kunnen worden aangepast voor domeinspecifieke data, zoals sonarafbeeldingen.

#### A.2.4. Optimalisatie van sonardata

Sonardata vormt een unieke uitdaging voor objectdetectie vanwege de specifieke kenmerken van akoestische beelden, zoals ruis, lage resolutie en reflecties. De meeste technieken zijn echter oorspronkelijk ontwikkeld voor visuele data. Er zijn dus verschillende aanpassingen nodig om deze methoden effectief te laten werken op sonarafbeeldingen. Één van deze aanpassingen zit hem in het pre-processen van de data met behulp van filters om ruis te onderdrukken en het contrast te verbeteren. Ook kan er gebruik gemaakt worden van aangepaste modellen die speciaal kunnen worden ontworpen om beter te werken met sonardata. Dit kan door rekening te houden met hoe objecten in de data ruimtelijk met elkaar verbonden zijn en met specifieke geluidspatronen die in sonarbeelden voorkomen. (Karimanzira e.a., 2020)

#### A.2.5. Formaat van de data

Daarnaast is het formaat van de data cruciaal. Er wordt (hoogstwaarschijnlijk) gebruik gemaakt van stacked GeoTIFF. Hierbij wordt het GeoTIFF formaat gebruikt om meerdere lagen geografische data te combineren in één bestand. GeoTIFF is een uitbreiding van het standaard TIFF (Tagged Image File Format) en wordt gebruikt om rasterafbeeldingen te koppelen aan geografische coördinaten. (Ritter & Ruth, 1997)

#### A.2.6. Uitdagingen & opportuniteiten

De vorige paragrafen bespraken vooral het technische aspect van semi-supervised en self-supervised learning. De algemene consensus is dat deze technieken veel-

belovend zijn om, bijvoorbeeld, een computervisie model te trainen. Een ander domein waar deze technieken zeer goed van pas komen, is NLP. Toch bestaan er enkele uitdagingen, al dan niet voor deze specifieke use-case:

- Modeloptimalisatie: sonarafbeeldingen vereisen aangepaste augmentaties en loss-functies.
- Dataschaal en kwaliteit: ongesuperviseerde sonardata kan ruis bevatten die het leerproces beïnvloedt.
- Prestatievergelijking: het vaststellen van de minimale hoeveelheid gelabelde data die nodig is om vergelijkbare resultaten te bereiken als volledig gesuperviseerde methoden.

### **A.2.7. Conclusie**

De literatuur toont aan dat semi- en self-supervised learning krachtige technieken zijn om de afhankelijkheid van handmatige labeling te verminderen. Door deze methoden aan te passen aan de unieke eigenschappen van sonardata, zoals ruis en textuurkenmerken, kan de efficiëntie van het labelproces aanzienlijk worden verbeterd. Dit biedt een veelbelovende richting voor verder onderzoek en praktische toepassingen in sonarbeeldanalyse.

## **A.3. Methodologie**

Dit onderzoek volgt een gestructureerde aanpak om semi- en self-supervised learning technieken te evalueren voor objectdetectie in sonardata. De methodologie is onderverdeeld in zes fasen: literatuurstudie, data pre-processing, modelontwikkeling, training en evaluatie.

### **A.3.1. Fase 1**

De eerste fase richt zich op het verzamelen van gedetailleerde informatie over het domein en de huidige stand van zaken hierbinnen. Specifiek gaat dit om het verzamelen, bestuderen en analyseren van wetenschappelijke artikelen. Op basis hiervan zal het onderzoek verder uitgewerkt worden.

### **A.3.2. Fase 2**

De tweede fase richt zich op het verzamelen, pre-processen en verdelen van de benodigde data voor dit onderzoek. Er wordt gebruik gemaakt van sonarafbeeldingen in gestapeld GeoTIFF-formaat. Deze data bevat verschillende lagen, zoals intensiteit en diepte-informatie, die dienen als input voor het model. Daarna zal pre-processing op de data toegepast worden. Er zal gebruik gemaakt worden van normalisatie, waarbij ruis en artefacten worden verminderd door technieken zoals

median filtering. Daarnaast zullen augmentaties zoals rotatie, ruisinjectie en schaling worden toegepast om variatie in de dataset te vergroten en robuustheid van het model te verbeteren. Uiteindelijk wordt de data opgedeeld in een gelabelde en een grotere ongesuperviseerde subset. Dit maakt het mogelijk om zowel semi- als self-supervised technieken te testen.

### **A.3.3. Fase 3**

De derde fase richt zich op de ontwikkeling van de modellen zelf. Een baseline-model zoals een convolutioneel neurale netwerk (bijvoorbeeld Faster R-CNN of YOLO) wordt gebruikt als referentiepunt voor volledig gesuperviseerde prestaties. Daarna worden self-supervised- of semi-supervised-modellen getraind worden ter vergelijking met het baseline-model. Binnen semi-supervised learning zal er geëxperimenteerd worden met pseudo-labeling: ongesuperviseerde voorbeelden worden automatisch gelabeld door het model en toegevoegd aan de trainingsset. Ook technieken zoals FixMatch om consistente voorspellingen te leren zullen bekeken en besproken worden. Binnen self-supervised learning zullen methoden zoals SimCLR of BYOL toegepast worden om representaties te leren zonder labels. Ook zullen aanpassingen aan pretext-taken, zoals het voorspellen van ontbrekende delen van sonarafbeeldingen of contrastieve augmentaties die rekening houden met spatiële afhankelijkheden besproken worden.

### **A.3.4. Fase 4**

De vierde fase richt zich op het trainen en optimaliseren van de ontwikkelde modellen. Dit omvat de pre-training, waarbij het model getraind wordt met ongesuperviseerde data om algemene representaties te leren. Daarnaast zal het model fine-tuning ondergaan: na pretraining wordt het model verder getraind met een kleine gelabelde dataset om objectdetectie te verfijnen. Ook hyperparameter-tuning behoort tot deze fase. Hierbij worden parameters zoals leersnelheid, batchgrootte en augmentatie-instellingen geoptimaliseerd om de prestaties te verbeteren.

### **A.3.5. Fase 5**

De vijfde fase richt zich op de evaluatie van de verschillende modellen. Er worden verschillende metrieken gebruikt om de modellen te evalueren. Onder andere Mean Average Precision (mAP) voor objectdetectieprestaties, Intersection over Union (IoU) voor lokalisatienauwkeurigheid en label-efficiëntie (Hoe goed presteert het model met een beperkte hoeveelheid gelabelde data?) zullen gebruikt worden. Er zal ook een vergelijking gemaakt worden met het baseline-model. Dit zal uitsluitend geven over de effectiviteit van semi-supervised en self-supervised learning tegenover supervised learning. Uiteindelijk zullen enkele robuustheidstests uitgevoerd worden. Het model wordt hierbij getest op ongeziene data met variërende omstandigheden, zoals ruisniveaus en objecttypes.

### **A.3.6. Fase 6**

De zesde fase richt zich op de validatie en praktische toepassing van het model. Hierbij worden de modellen getest op een onafhankelijke dataset van sonarafbeeldingen om generaliseerbaarheid te evalueren. Ook zullen enkele experts in sonaranalyse de bruikbaarheid van de resultaten beoordelen en aanbevelingen geven voor verdere verbeteringen. Ten slotte zullen de methodologie, resultaten en code worden gedocumenteerd om reproduceerbaarheid te waarborgen.

### **A.4. Verwacht resultaat, conclusie**

Het onderzoek verwacht aan te tonen dat semi- en self-supervised learning effectief kunnen worden ingezet om het labelproces bij sonarobjectdetectie aanzienlijk te versnellen. Door technieken zoals SimCLR of BYOL te gebruiken voor pre-training, wordt verwacht dat het model sterke representaties leert van unsupervised sonar data, wat de behoefte aan grootschalige gelabelde datasets verkleint. Daarnaast zal een analyse inzicht geven in de minimale hoeveelheid gelabelde data die nodig is om vergelijkbare of betere prestaties te behalen dan met volledig supervised-learning methoden. Dit resulteert in een efficiëntere en kosteneffectieve aanpak voor objectdetectie in sonarbeelden, zonder verlies van nauwkeurigheid, en biedt een waardevolle methodologie voor verdere toepassingen in domeinen waar gelabelde data schaars is.

# Bibliografie

- Adaloglou, N. (2022, mei 12). *BYOL tutorial: self-supervised learning on CIFAR images with code in Pytorch*. <https://theaisummer.com/byol/>
- Álvarez-Tuñón, O., Marnet, L. R., Antal, L., Aubard, M., Costa, M., & Brodskiy, Y. (2024). SubPipe: A Submarine Pipeline Inspection Dataset for Segmentation and Visual-inertial Localization. <https://doi.org/10.48550/ARXIV.2401.17907>
- Anish Dev, J. (2014). Bitcoin mining acceleration and performance quantification. *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1–6. <https://doi.org/10.1109/ccece.2014.6900989>
- Aridgides, T., Antoni, D., Fernandez, M. F., & Dobeck, G. J. (1995, juni). Adaptive filter for mine detection and classification in side-scan sonar imagery. In A. C. Dubey, I. Cindrich, J. M. Ralston & K. A. Rigano (Red.), *Detection Technologies for Mines and Minelike Targets* (pp. 475–486, Deel 2496). SPIE. <https://doi.org/10.1117/12.211345>
- Aubard, M., Antal, L., Madureira, A., Teixeira, L. F., & Ábrahám, E. (2024). ROSAR: An Adversarial Re-Training Framework for Robust Side-Scan Sonar Object Detection. <https://doi.org/10.48550/ARXIV.2410.10554>
- Aubard, M., Antal, L., Madureira, M., F. Teixeira, L., & Ábrahám, E. (2024). SWDD: Sonar Wall Detection Dataset. <https://doi.org/10.5281/ZENODO.10528134>
- Aubard, M., Madureira, A., Teixeira, L., & Pinto, J. (2024). Sonar-based Deep Learning in Underwater Robotics: Overview, Robustness and Challenges. <https://doi.org/10.48550/ARXIV.2412.11840>
- Awalludin, E. A., Arsad, T. N. T., Yussof, W. N. J. H. W., Bachok, Z., & Hitam, M. S. (2022). A Comparative Study of Various Edge Detection Techniques for Underwater Images. *Journal of Telecommunications and Information Technology*, 1(2022), 23–33. <https://doi.org/10.26636/jtit.2022.155921>
- Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., & Raffel, C. A. (2019). MixMatch: A Holistic Approach to Semi-Supervised Learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox & R. Garnett (Red.), *Advances in Neural Information Processing Systems* (Deel 32). Curran Associates, Inc. [https://proceedings.neurips.cc/paper\\_files/paper/2019/file/1cd138d0499a68f4bb72bee04bbec2d7-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/1cd138d0499a68f4bb72bee04bbec2d7-Paper.pdf)
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. <https://doi.org/10.48550/ARXIV.2004.10934>
- Bourke, P. (1998). Bmp image format. *BMP Files*. July, 8.

- C A Padmanabha Reddy, Y., Viswanath, P., & Eswara Reddy, B. (2018). Semi-supervised learning: a brief review. *International Journal of Engineering & Technology*, 7(1.8), 81. <https://doi.org/10.14419/ijet.v7i1.8.9977>
- Carranza-García, M., Torres-Mateo, J., Lara-Benítez, P., & García-Gutiérrez, J. (2020). On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data. *Remote Sensing*, 13(1), 89. <https://doi.org/10.3390/rs13010089>
- Cascante-Bonilla, P., Tan, F., Qi, Y., & Ordonez, V. (2020). Curriculum Labeling: Revisiting Pseudo-Labeling for Semi-Supervised Learning. <https://doi.org/10.48550/ARXIV.2001.06001>
- Chen, C., Liu, M.-Y., Tuzel, O., & Xiao, J. (2017). R-CNN for Small Object Detection. In *Computer Vision – ACCV 2016* (pp. 214–230). Springer International Publishing. [https://doi.org/10.1007/978-3-319-54193-8\\_14](https://doi.org/10.1007/978-3-319-54193-8_14)
- Chen, C., Zheng, Z., Xu, T., Guo, S., Feng, S., Yao, W., & Lan, Y. (2023). YOLO-Based UAV Technology: A Review of the Research and Its Applications. *Drones*, 7(3), 190. <https://doi.org/10.3390/drones7030190>
- Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A Simple Framework for Contrastive Learning of Visual Representations. <https://doi.org/10.48550/ARXIV.2002.05709>
- Dahn, N., Bande Firvida, M., Sharma, P., Mohrmann, J., Geisler, O., Sanghamreddy, P. K., Marquardt, K., & Christensen, L. (2024). An Acoustic and Optical Dataset for the Perception of Underwater Unexploded Ordnance (UXO). <https://doi.org/10.5281/ZENODO.11068045>
- Dahn, N., Firvida, M. B., Sharma, P., Christensen, L., Geisle, O., Mohrmann, J., Frey, T., Kumar Sanghamreddy, P., & Kirchner, F. (2024). An Acoustic and Optical Dataset for the Perception of Underwater Unexploded Ordnance (UXO). *OCEANS 2024 - Halifax*, 1–6. <https://doi.org/10.1109/oceans55160.2024.10754316>
- DeVries, T., & Taylor, G. W. (2017). Improved Regularization of Convolutional Neural Networks with Cutout. <https://doi.org/10.48550/ARXIV.1708.04552>
- Dimitrova-Grekow, T., Salauyou, V., & Kowalski, K. (2017). Indoor Mapping Using Sonar Sensor and Otsu Method. *Measurement Automation Monitoring*, 63(6), 214–216. <https://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-c104553b-2ef9-4d60-85b1-8d7623944a7d>
- Ding, L., & Goshtasby, A. (2001). On the Canny edge detector. *Pattern Recognition*, 34(3), 721–725. [https://doi.org/10.1016/s0031-3203\(00\)00023-6](https://doi.org/10.1016/s0031-3203(00)00023-6)
- Diwan, T., Anirudh, G., & Tembhurne, J. V. (2022). Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, 82(6), 9243–9275. <https://doi.org/10.1007/s11042-022-13644-y>

- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2009). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303–338. <https://doi.org/10.1007/s11263-009-0275-4>
- Fan, Y., Kukleva, A., Dai, D., & Schiele, B. (2022). Revisiting Consistency Regularization for Semi-Supervised Learning. *International Journal of Computer Vision*, 131(3), 626–643. <https://doi.org/10.1007/s11263-022-01723-4>
- Ferreira, R. E. P., Lee, Y. J., & Dórea, J. R. R. (2023). Using pseudo-labeling to improve performance of deep neural networks for animal identification. *Scientific Reports*, 13(1). <https://doi.org/10.1038/s41598-023-40977-x>
- Géron, A. (2023). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (Third edition) [Literaturangaben. - Index]. O'Reilly.
- Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. <https://doi.org/10.48550/ARXIV.1311.2524>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* (A. Courville & Y. Bengio, Red.) [Includes bibliographical references and index]. The MIT Press.
- Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., & Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised Learning. <https://doi.org/10.48550/ARXIV.2006.07733>
- Gui, J., Chen, T., Zhang, J., Cao, Q., Sun, Z., Luo, H., & Tao, D. (2024). A Survey on Self-Supervised Learning: Algorithms, Applications, and Future Trends. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12), 9052–9071. <https://doi.org/10.1109/tpami.2024.3415112>
- Gupta, K., Ajanthan, T., Hengel, A. v. d., & Gould, S. (2022). Understanding and Improving the Role of Projection Head in Self-Supervised Learning. <https://doi.org/10.48550/ARXIV.2212.11491>
- Hady, M. F. A., & Schwenker, F. (2013). Semi-supervised Learning. In *Handbook on Neural Information Processing* (pp. 215–239). Springer Berlin Heidelberg. [https://doi.org/10.1007/978-3-642-36657-4\\_7](https://doi.org/10.1007/978-3-642-36657-4_7)
- Hall, P. (1987). On Kullback-Leibler Loss and Density Estimation. *The Annals of Statistics*, 15(4), 1491–1519. <https://doi.org/10.1214/aos/1176350606>
- He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2019). Momentum Contrast for Unsupervised Visual Representation Learning. <https://doi.org/10.48550/ARXIV.1911.05722>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. <https://doi.org/10.1109/cvpr.2016.90>

- <http://www.freephotos.lu/>. (2010, juni 14). *Image processing illustration, before Otsu algorithm*. [http://en.wikipedia.org/wiki/File:Image\\_processing\\_pre\\_otsus\\_algorithm.jpg](http://en.wikipedia.org/wiki/File:Image_processing_pre_otsus_algorithm.jpg)
- Imambi, S., Prakash, K. B., & Kanagachidambaresan, G. R. (2021). PyTorch. In *Programming with TensorFlow* (pp. 87–104). Springer International Publishing. [https://doi.org/10.1007/978-3-030-57077-4\\_10](https://doi.org/10.1007/978-3-030-57077-4_10)
- Jian, Y., & Kaibing, X. (2022). Underwater acoustic target detection (UATD) dataset. <https://doi.org/10.6084/M9.FIGSHARE.21331143.V3>
- Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A Review of Yolo Algorithm Developments. *Procedia Computer Science*, 199, 1066–1073. <https://doi.org/10.1016/j.procs.2022.01.135>
- Jiang, Z., & Wang, R. (2020). Underwater Object Detection Based on Improved Single Shot MultiBox Detector. *2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence*, 1–7. <https://doi.org/10.1145/3446132.3446170>
- Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., Boyle, R., Cantin, P.-I., Chao, C., Clark, C., Coriell, J., Daley, M., Dau, M., Dean, J., Gelb, B., ... Yoon, D. H. (2017). In-Datacenter Performance Analysis of a Tensor Processing Unit. *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 1–12. <https://doi.org/10.1145/3079856.3080246>
- Kage, P., Rothenberger, J. C., Andreadis, P., & Diochnos, D. I. (2024). A Review of Pseudo-Labeling for Computer Vision. <https://doi.org/10.48550/ARXIV.2408.07221>
- Karimanzira, D., Renkewitz, H., Shea, D., & Albiez, J. (2020). Object Detection in Sonar Images. *Electronics*, 9(7), 1180. <https://doi.org/10.3390/electronics9071180>
- Komari Alaie, H., & Farsi, H. (2018). Passive Sonar Target Detection Using Statistical Classifier and Adaptive Threshold. *Applied Sciences*, 8(1), 61. <https://doi.org/10.3390/app8010061>
- Kumar, A., & Srivastava, S. (2020). Object Detection System Based on Convolution Neural Networks Using Single Shot Multi-Box Detector. *Procedia Computer Science*, 171, 2610–2617. <https://doi.org/10.1016/j.procs.2020.04.283>
- Lee, D.-H. (2013). Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*.
- Lee, S., Park, B., & Kim, A. (2018). Deep Learning from Shallow Dives: Sonar Image Generation and Training for Underwater Object Detection. <https://doi.org/10.48550/ARXIV.1810.07990>

- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2015). SSD: Single Shot MultiBox Detector, 21–37. [https://doi.org/10.1007/978-3-319-46448-0\\_2](https://doi.org/10.1007/978-3-319-46448-0_2)
- Long, J., Shelhamer, E., & Darrell, T. (2015). Fully Convolutional Networks for Semantic Segmentation. <https://doi.org/10.1109/cvpr.2015.7298965>
- Loshchilov, I., & Hutter, F. (2016). SGDR: Stochastic Gradient Descent with Warm Restarts. <https://doi.org/10.48550/ARXIV.1608.03983>
- Lourey, S. (2017). Adaptive filtering for enhanced detection of continuous active sonar signals. *Proc. Underwater Acoust. Conf. Exhib.(UACE)*, 145–152. [https://www.uaconferences.org/docs/2017\\_papers/153\\_UACE2017.pdf](https://www.uaconferences.org/docs/2017_papers/153_UACE2017.pdf)
- Lu, Z., Wang, J., & Song, J. (2019). Multi-resolution CSI Feedback with deep learning in Massive MIMO System. <https://doi.org/10.48550/ARXIV.1910.14322>
- Lucas, T., Weinzaepfel, P., & Rogez, G. (2022). Barely-Supervised Learning: Semi-supervised Learning with Very Few Labeled Images. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(2), 1881–1889. <https://doi.org/10.1609/aaai.v36i2.20082>
- Ma, W., Wang, X., & Yu, J. (2020). A Lightweight Feature Fusion Single Shot Multibox Detector for Garbage Detection. *IEEE Access*, 8, 188577–188586. <https://doi.org/10.1109/access.2020.3031990>
- Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., & Wu, H. (2017). Mixed Precision Training. <https://doi.org/10.48550/ARXIV.1710.03740>
- Min, Z., Ge, Q., & Tai, C. (2022). Why the pseudo label based semi-supervised learning algorithm is effective? <https://doi.org/10.48550/ARXIV.2211.10039>
- Murray-Rust, P. (2008). Open Data in Science. *Nature Precedings*. <https://doi.org/10.1038/npre.2008.1526.1>
- Oord, A. v. d., Li, Y., & Vinyals, O. (2018). Representation Learning with Contrastive Predictive Coding. <https://doi.org/10.48550/ARXIV.1807.03748>
- Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62–66. <https://doi.org/10.1109/tsmc.1979.4310076>
- Pang, B., Nijkamp, E., & Wu, Y. N. (2019). Deep Learning With TensorFlow: A Review. *Journal of Educational and Behavioral Statistics*, 45(2), 227–248. <https://doi.org/10.3102/1076998619872761>
- Pessanha Santos, N., & Moura, R. (2024). Side-scan sonar imaging for Mine detection. <https://doi.org/10.6084/M9.FIGSHARE.24574879>
- Pessanha Santos, N., Moura, R., Sampaio Torgal, G., Lobo, V., & Neto, M. d. C. (2024). Side-scan sonar imaging data of underwater vehicles for mine detection. *Data in Brief*, 53, 110132. <https://doi.org/10.1016/j.dib.2024.110132>

- Pikez33. (2010, juni 14). *Image processing illustration, after Otsu algorithm*. [https://commons.wikimedia.org/wiki/File:Image\\_processing\\_post\\_otsus\\_algorithm.jpg](https://commons.wikimedia.org/wiki/File:Image_processing_post_otsus_algorithm.jpg)
- Poskanzer, J. (2016, oktober 9). *pgm - Netpbm grayscale image format*. <https://netpbm.sourceforge.net/doc/pgm.html>
- Prechelt, L. (1998). Early Stopping - But When? In *Neural Networks: Tricks of the Trade* (pp. 55–69). Springer Berlin Heidelberg. [https://doi.org/10.1007/3-540-49430-8\\_3](https://doi.org/10.1007/3-540-49430-8_3)
- Priyadharsini, R., & Sharmila, T. S. (2019). Object Detection In Underwater Acoustic Images Using Edge Based Segmentation Method. *Procedia Computer Science*, 165, 759–765. <https://doi.org/10.1016/j.procs.2020.01.015>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection, 779–788. <https://doi.org/10.1109/cvpr.2016.91>
- Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. <https://doi.org/10.48550/ARXIV.1612.08242>
- Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. <https://doi.org/10.48550/ARXIV.1804.02767>
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. <https://doi.org/10.48550/ARXIV.1506.01497>
- Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. <https://doi.org/10.48550/ARXIV.1902.09630>
- Ritter, N., & Ruth, M. (1997). The GeoTiff data interchange standard for raster geographic images. *International Journal of Remote Sensing*, 18(7), 1637–1647. <https://doi.org/10.1080/014311697218340>
- Rosebrock, A. (2016, november 7). *Intersection over Union (IoU) for object detection*. <https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>
- Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., & Raffel, C. (2020). FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. <https://doi.org/10.48550/arXiv.2001.07685>
- Song, Z., Yang, X., Xu, Z., & King, I. (2021). Graph-based Semi-supervised Learning: A Comprehensive Review. <https://doi.org/10.48550/ARXIV.2102.13303>
- soundmetrics.com. (g.d.). *ARIS EXPLORER 3000*. <http://www.soundmetrics.com/products/aris-sonars/aris-explorer-3000>
- Sowe, E. A., & Bah, Y. A. (2025). Momentum Contrast for Unsupervised Visual Representation Learning. *Journal of Advances in Civil and Mechanical Engineering*. <https://doi.org/10.20944/preprints202501.0668.v1>

- TechPowerUp. (g.d.-a). NVIDIA RTX A5000. <https://www.techpowerup.com/gpu-specs/rtx-a5000.c3748>
- TechPowerUp. (g.d.-b). NVIDIA Tesla T4. <https://www.techpowerup.com/gpu-specs/tesla-t4.c3316>
- Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*, 5(4), 1680–1716. <https://doi.org/10.3390/make5040083>
- Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient Object Localization Using Convolutional Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. <https://doi.org/10.1109/cvpr.2015.7298664>
- Torre, V., & Poggio, T. A. (1986). On Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(2), 147–163. <https://doi.org/10.1109/tpami.1986.4767769>
- Valdenegro-Toro, M. (2019). Learning Objectness from Sonar Images for Class-Independent Object Detection. <https://doi.org/10.48550/ARXIV.1907.00734>
- van Engelen, J. E., & Hoos, H. H. (2019). A survey on semi-supervised learning. *Machine Learning*, 109(2), 373–440. <https://doi.org/10.1007/s10994-019-05855-6>
- van Laarhoven, T. (2017). L2 Regularization versus Batch and Weight Normalization. <https://doi.org/10.48550/ARXIV.1706.05350>
- Wang, B. (2022). A Parallel Implementation of Computing Mean Average Precision. <https://doi.org/10.48550/ARXIV.2206.09504>
- Wang, H., & Xiao, N. (2023). Underwater Object Detection Method Based on Improved Faster RCNN. *Applied Sciences*, 13(4), 2746. <https://doi.org/10.3390/app13042746>
- Wang, W., Zhang, Q., Qi, Z., & Huang, M. (2024). CenterNet-Saccade: Enhancing Sonar Object Detection with Lightweight Global Feature Extraction. *Sensors*, 24(2), 665. <https://doi.org/10.3390/s24020665>
- Wang, Y. E., Wei, G.-Y., & Brooks, D. (2019). Benchmarking TPU, GPU, and CPU Platforms for Deep Learning. <https://doi.org/10.48550/ARXIV.1907.10701>
- Xie, K., Yang, J., & Qiu, K. (2022). A Dataset with Multibeam Forward-Looking Sonar for Underwater Object Detection. *Scientific Data*, 9(1). <https://doi.org/10.1038/s41597-022-01854-w>
- Ying, X. (2019). An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*, 1168, 022022. <https://doi.org/10.1088/1742-6596/1168/2/022022>
- Yuan, X., Martínez, J.-F., Eckert, M., & López-Santidrián, L. (2016). An Improved Otsu Threshold Segmentation Method for Underwater Simultaneous Localization and Mapping-Based Navigation. *Sensors*, 16(7), 1148. <https://doi.org/10.3390/s16071148>

- Yulin, T., Shaohua, J., Gang, B., Yonzhou, Z., & Fan, L. (2020). Wreckage Target Recognition in Side-scan Sonar Images Based on an Improved Faster R-CNN Model. *2020 International Conference on Big Data amp; Artificial Intelligence amp; Software Engineering (ICBASE)*, 348–354. <https://doi.org/10.1109/icbase51474.2020.00080>
- Zeng, L., Sun, B., & Zhu, D. (2021). Underwater target detection based on Faster R-CNN and adversarial occlusion network. *Engineering Applications of Artificial Intelligence*, 100, 104190. <https://doi.org/10.1016/j.engappai.2021.104190>
- Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2017). mixup: Beyond Empirical Risk Minimization. <https://doi.org/10.48550/ARXIV.1710.09412>
- Zhou, P., Xie, X., Lin, Z., & Yan, S. (2024). Towards Understanding Convergence and Generalization of AdamW. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(9), 6486–6493. <https://doi.org/10.1109/tpami.2024.3382294>
- Zhu, X. J. (2005). Semi-Supervised Learning Literature Survey. *CS Technical Reports*.