HAN_UNIVERSITY
OF APPLIED SCIENCES

ONDERZOEKSVERSLAG OBJECT DETECTIE_

NOTS

Groep 3 Opleiding: HBO-ICT Studiejaar: 3 31 januari 2023 Docenten: Martijn Driessen Versie 2.1	**Studenten:** Sven van Ee 1645999 Sam Wolters 1653733 Marijn Martin 1547235 Sjoerd de Bruin 1650151

HAN_UNIVERSITY
OF APPLIED SCIENCES

SAMENVATTING

Dit is een onderzoeksverslag voor het NotS-project. Voor het NotS-project moet er een applicatie gerealiseerd worden die objecten in afbeeldingen en video's kan detecteren en classificeren. Om dit doel te bereiken moeten er verschillende onderzoeken gedaan worden naar geschikte technieken en algoritmen voor objectdetectie en classificatie. In dit onderzoek beantwoorden we de hoofdvraag "Wat is object detectie en welk algoritme is het meest geschikt voor het NotS-project", en dit zal worden aangepakt door verschillende subvragen te beantwoorden met betrekking tot objectdetectietechnologie, technieken, voor- en nadelen, factoren die bijdragen aan de nauwkeurigheid van objectdetectie en de verschillen en overeenkomsten tussen verschillende objectdetectiealgoritmen.

Met opmerkingen [SW(1]: Te vaag. Duidelijk maken wat het project is

Met opmerkingen [SW(2R1]: Mag ook opgedeeld worden in 2 hoofdvragen.

HAN_UNIVERSITY
OF APPLIED SCIENCES

INHOUDSOPGAVE

INLEIDING

Tegenwoordig worden computers steeds vaker gebruikt om complexe taken uit te voeren die voorheen alleen door mensen konden worden uitgevoerd. Object detectie is zo'n taak waarin computers steeds beter presteren. Object detectie is een techniek waarmee computers automatisch objecten in afbeeldingen en video's kunnen herkennen, wat kan worden gebruikt voor verschillende toepassingen, zoals autonome voertuigen, bewakingssystemen en gezichtsherkenning.

Het NotS-project richt zicht op het ontwikkelen van een applicatie waarin object detectie kan worden toegepast. De applicatie moet in staat zijn om objecten in afbeeldingen en video's te detecteren en te classificeren. Binnen dit onderzoek wordt onderzocht hoe object detectie werkt en welke algoritmes het beste van toepassing zijn bij het maken van de applicatie voor het NotS-project.

Om deze vragen te beantwoorden, wordt in dit onderzoek onderzocht wat object detectie is en welke algoritmes momenteel het meest geschikt zijn voor het NotS-project. Daarnaast wordt er gekeken naar de verschillende technieken die worden gebruikt voor object detectie en de factoren die bijdragen aan de nauwkeurigheid en snelheid van een object detectiemodel. Door deze vragen te beantwoorden, hopen we een beter begrip te krijgen van object detectie en hoe deze technologie gebruikt kan worden binnen de applicatie.

De resultaten van dit onderzoek zullen bijdragen aan de verdere ontwikkeling van de applicatie voor het NotS-project. Daarnaast zal het onderzoek bijdragen aan de kennis over object detectie en kunnen de resultaten gebruikt worden voor toekomstige onderzoeksprojecten op dit gebied. De rest van het onderzoek zal worden georganiseerd volgens de volgende structuur: allereerst wordt er gekeken naar de definitie van objectie, daarna wordt er gekeken naar de verschillende technieken die momenteel worden gebruikt voor object detectie. Vervolgens wordt er gekeken naar de factoren die bijdragen aan de nauwkeurigheid van een object detectiemodel en tenslotte worden er verschillende algoritmes voor object detectie met elkaar vergeleken om te bepalen welk algoritme het meest geschikt is voor het NotS-project.

1 PROBLEEMSTELLING

Voor het NotS-project moet er een applicatie ontwikkelt worden waarin objecten in afbeeldingen en video's gedetecteerd en geclassificeerd worden. Om deze applicatie te kunnen realiseren, moet er onderzoek gedaan worden naar de verschillende technieken en algoritmes voor object detectie. Het is van belang om dit uit te zoeken, omdat er momenteel geen kennis binnen de projectgroep is over het detecteren van objecten in afbeeldingen en het werken met deep-learning.

De hoofdvraag voor dit onderzoek luidt als volgt: "Wat is object detectie en welk algoritme is het meest geschikt voor het NotS-project?". Om deze hoofdvraag te beantwoorden, zullen de volgende deelvragen worden behandeld:

- Wat is object detectie?
- Hoe werkt object detectie?
- Wat zijn de verschillen en overeenkomsten tussen de verschillende algoritmes voor object detectie?
- Wat zijn de voor- en nadelen van object detectie technieken?
- Welke factoren dragen bij aan de nauwkeurigheid van een object detectiemodel?

Door deze deelvragen te beantwoorden, zal er meer inzicht worden verkregen in de technologie van object detectie en welk algoritme het best ingezet kan worden voor het NotS-project. Hiermee kan er een effectieve applicatie ontwikkeld worden die objecten in afbeeldingen en video's detecteert en classificeert.

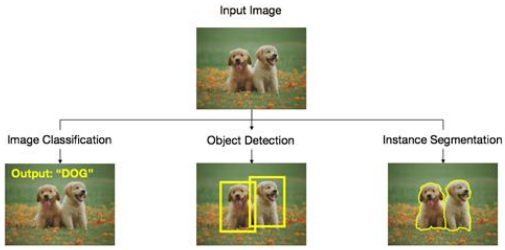
2 DOELSTELLING

Het doel van dit onderzoek is om een beter begrip te krijgen van object detectie en welk algoritme het meest geschikt is voor het NotS-project. Dit doel zal bereikt worden door de volgende doelstellingen te behalen:

1. Het verkrijgen van een duidelijk begrip van object detectie en de werking van verschillende technieken voor object detectie.
2. Het verkrijgen van inzicht in de verschillende factoren die bijdragen aan de nauwkeurigheid van een object detectiemodel en hoe deze factoren geoptimaliseerd kunnen worden.
3. Het vergelijken van de prestaties van verschillende algoritmes, om te bepalen welk algoritme het meest geschikt is voor het NotS-project.

3 WAT IS OBJECT DETECTIE?

Object detectie is een computervisie techniek die het mogelijk maakt om automatisch objecten in afbeeldingen of video's te identificeren en lokaliseren. Het doel van object detectie is om een model te creëren dat in staat is om de aanwezigheid van een specifiek object in een afbeelding of video te detecteren en te classificeren, en tegelijkertijd de locatie van dat object in de afbeelding of video te bepalen.



Figuur 1: Image classification vs object detection vs instance segmentation (Loy, 2019)

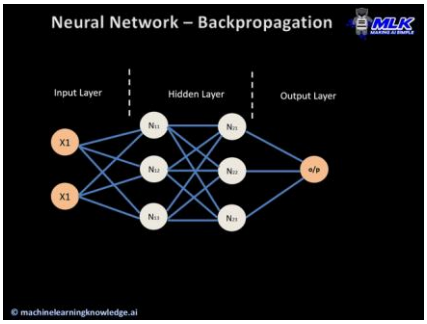
Het proces van object detectie kan worden onderverdeeld in twee belangrijke onderdelen: classificatie en lokalisatie. Classificatie verwijst naar het proces waarbij het model bepaalt tot welke klasse het gedetecteerde object behoort, zoals bijvoorbeeld een persoon, een auto of een hond. Lokalisatie verwijst daarentegen naar het proces waarbij het model de locatie van het object in de afbeelding of video bepaalt. Dit gebeurt meestal in vorm van een begrenzingsdoos (bounding box).

Figuur 2: Object detection algoritme (The MathWorks, Inc, sd)

Om dit te kunnen realiseren maakt object detectie gebruik van deep learning technieken, zoals convolutionele neurale netwerken (CNN's) en recurrente neurale netwerken (RNN's). Een CNN is in staat om afbeeldingskenmerken te extraheren door middel van convolutie, pooling en activatie

operaties. Terwijl een RNN in staat is om contextuele informatie te begrijpen door middel van het verwerken van sequentiële gegevens, zoals videoframes.

Het trainen van een object detectie model vereist meestal een grote hoeveelheid gelabelde data. Dit zijn afbeeldingen of video's waarbij elk object dat moet worden gedetecteerd handmatig is geannoteerd met de juiste klasse en locatie. Het model wordt getraind door middel van backpropagation, een algoritme dat de fout tussen de voorspelde output en de werkelijke output minimaliseert. Na het trainen van het model kan het worden gebruikt om nieuwe afbeeldingen en video's te analyseren en te detecteren of er objecten aanwezig zijn.



Figuur 3: Werking backpropagation (Goglia, 2021)

3.1 Convolutionele neurale netwerken

Convolutionele neurale netwerken (CNN's) zijn een soort neurale netwerken die vaak worden gebruikt voor beeld- en videoherkenning. Ze zijn ontworpen om het menselijke visuele systeem na te bootsen en te leren hoe ze afbeeldingen kunnen herkennen en classificeren.

Een CNN werkt door lagen van filters toe te passen op een afbeelding, waarbij elke filter een specifieke eigenschap van de afbeelding detecteert, zoals randen, hoeken en texturen. Door deze filters toe te passen en vervolgens de resulterende uitvoer door te geven aan volgende lagen, kan een CNN complexe kenmerken van de afbeelding leren en uiteindelijk een classificatie maken op basis van wat het heeft geleerd.

CNN's hebben een aantal voordelen ten opzichte van traditionele methoden voor beeldherkenning, omdat ze robuuster zijn tegen variaties in de afbeelding, zoals veranderingen in verlichting en schaal. Ze hebben een breed scala aan toepassingen, zoals beeldclassificatie, objectdetectie, gezichtsherkenning en zelfrijdende auto's.

3.2 Recurrente neurale netwerken

Recurrente neurale netwerken (RNN's) kunnen ook worden gebruikt voor objectdetectie, hoewel ze minder vaak worden gebruikt dan andere architecturen zoals convolutionele neurale netwerken (CNN's). Een populaire aanpak voor objectdetectie met RNN's is de zogenaamde "Recurrent YOLO" (You Only Look Once) methode.

Recurrent YOLO is een uitbreiding van de YOLO-objectdetectiemethode die gebruikmaakt van een convolutioneel neuraal netwerk om objecten in afbeeldingen te detecteren. Bij Recurrent YOLO wordt een RNN gebruikt om de output van het convolutionele netwerk te verfijnen en aan te passen, zodat het beter in staat is om objecten te detecteren die moeilijk te onderscheiden zijn of deels worden verborgen.

De RNN kan de uitvoer van het convolutionele netwerk aanpassen door het te gebruiken als input voor elke tijdstap in de RNN-lus. Door informatie over de afbeelding op te slaan en te combineren van eerdere tijdstappen, kan de RNN betere en nauwkeurigere voorspellingen doen over de locaties en afmetingen van objecten in de afbeelding.

Over het algemeen is het gebruik van RNN's voor objectdetectie minder gangbaar dan het gebruik van CNN's. Dit komt omdat CNN's over het algemeen beter presteren in het leren van kenmerken van afbeeldingen en het detecteren van objecten.

3.3 Backpropagation

Backpropagation (ook wel bekend als backwards propagation of error) is een algoritme voor het trainen van neurale netwerken door het berekenen van de afgeleiden van de foutfunctie ten opzichte van de gewichten van het netwerk. Het algoritme werkt door het doorgeven van de fouten van de uitvoerlaag naar de verborgen lagen en vervolgens naar de invoerlaag van het netwerk, waarbij de gewichten van elke laag worden bijgewerkt op basis van hun bijdrage aan de fout.

Gewichten zijn de parameters in een neuraal netwerk die de sterkte van de verbindingen tussen de neuronen bepalen. In een neuraal netwerk worden signalen van de input laag doorgegeven aan de volgende lagen van het netwerk via de verbindingen tussen de neuronen. Elke verbinding heeft een bijbehorend gewicht dat de sterkte van de verbinding bepaalt (Duursma, n.d.).

Het algoritme maakt gebruik van de kettingregel van de differentiatie om de afgeleiden van de foutfunctie te berekenen ten opzichte van elk gewicht in het netwerk. Het begint met het berekenen van de fout tussen de uitvoer van het netwerk en de gewenste uitvoer (de zogenaamde "loss"). Vervolgens worden de fouten van de uitvoerlaag teruggegeven aan de verborgen lagen, waarbij de afgeleiden van de activatiefuncties worden berekend en de fouten worden gewogen en doorgegeven aan de vorige lagen.

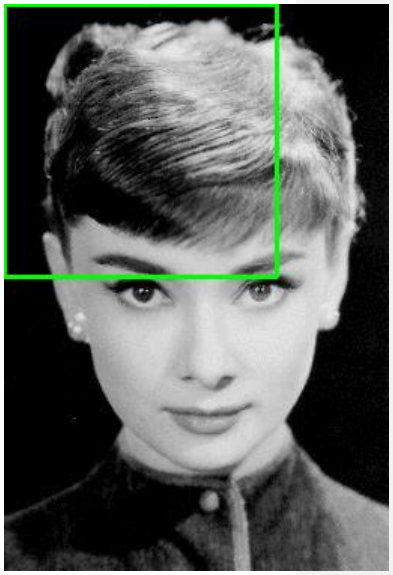
Uiteindelijk worden de gewichten van het netwerk bijgewerkt door de afgeleiden van de foutfunctie ten opzichte van de gewichten te berekenen en deze te gebruiken om de gewichten aan te passen om de fout te verminderen. Dit proces wordt herhaald totdat het netwerk de gewenste prestaties bereikt.

3.4 Sliding windows

Sliding windows is een veelgebruikte techniek bij object detectie om objecten in afbeeldingen te detecteren. Het werkt door een vast formaatvenster over de afbeelding te schuiven en vervolgens classificatie uit te voeren op elk venster om te bepalen of het een object bevat of niet.

Het formaat van het venster wordt meestal ingesteld op de grootte van het object dat wordt gezocht. Als het object bijvoorbeeld klein is, wordt het venster ook klein gemaakt en als het object groot is, wordt het venster vergroot.

Om de classificatie uit te voeren, wordt meestal een machine learning-model zoals een CNN gebruikt om te bepalen of het venster een object bevat of niet. Het model leert op basis van een dataset met afbeeldingen waarbij de labels zijn aangegeven welke delen van de afbeelding overeenkomen met een object en welke niet.

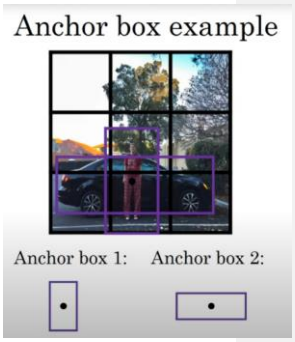


Figuur 4: Sliding windows werking (Rosebrock, 2015)

Sliding windows kunnen echter zeer rekenkundig intensief zijn, vooral bij het zoeken naar objecten op hoge resolutie afbeeldingen of het detecteren van meerdere objecten tegelijkertijd. Er zijn daarom andere technieken die worden gebruikt om objecten te detecteren, zoals region proposal networks en single-shot detectors.

3.5 Anchor boxes

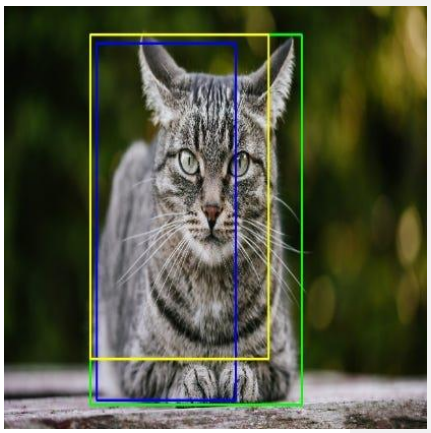
Een van de problemen met object detectie is dat een enkele grid cel, maar één object kan detecteren. Anchor boxes is een methode waarbij er meerdere objecten in een cel zitten kunnen gedetecteerd worden.
Anchor boxes zijn een set van vooraf gedefinieerde begrenzingsvakjes van een bepaalde hoogte en breedte. Deze vakjes zijn gedefinieerd om de schaal en aspectverhouding van specifieke objectklassen vast te leggen en worden typisch gekozen op basis van objectgroottes in uw trainingsdatasets. Het doel is om de vakjes zo te kiezen dat ze de objecten in de afbeelding zo goed mogelijk omvatten. Het neuraal netwerk voorspelt vervolgens de waarschijnlijkheid van elk vakje dat het een object bevat en de afwijking van het vakje van het object (The MathWorks, Inc., n.d.).
Anchor boxes worden gebruikt met onder andere YOLO en SSD.



Figuur 5: Anchor boxes (DeepLearningAI, 2017)

3.6 Non-max supression

Non-maximum suppression (NMS) is een computer vision-methode die een enkele entiteit selecteert uit veel overlappende entiteiten. In objectdetectie wordt non-maximum suppression gebruikt om de begrenzingsvakjes te filteren die een hoge waarschijnlijkheid hebben om een object te bevatten. Het algoritme selecteert de begrenzingsvakjes met de hoogste waarschijnlijkheid en onderdrukt vervolgens alle andere vakjes die een hoge overlap hebben met de geselecteerde vakjes. Zo moet in ***Fout! Verwijzingsbron niet gevonden.*** alleen het groene gebied overblijven, nadat non-max supression is gebruikt (Subramanyam, 2021).

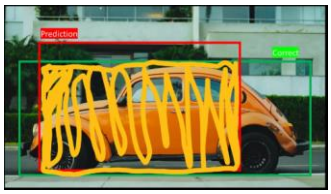


Figuur 6: Non-max supression kat (Subramanyam, 2021)

3.7 Intersection over Union

Intersection over Union (IoU) is een veelgebruikte metriek in computervisie om de nauwkeurigheid van objectdetectie te evalueren. IoU wordt berekend als de verhouding tussen de intersectie en de unie van twee gebieden in een afbeelding.
Om IoU te berekenen, worden twee gebieden vergeleken, bijvoorbeeld de voorspelde detectiebox en de werkelijke annotatie van het object. De intersectie is het gebied dat beide boxen overlappen, terwijl de unie het gebied omvat dat door beide boxen wordt afgedekt. De IoU wordt berekend door de intersection te delen door de union van de twee boxen.

Figuur 8: Voorspelde vs correcte bounding box (Persson, 2020)



Figuur 9: Intersection voorbeeld (Persson, 2020)

Figuur 7: Union voorbeeld (Persson, 2020)

Een IoU-waarde van 1 betekent dat de twee boxen exact samenvallen, terwijl een IoU-waarde van 0 betekent dat de boxen geen overlap hebben. In objectdetectie wordt vaak een drempelwaarde voor de IoU ingesteld om te bepalen wanneer een detectie als een correcte detectie wordt beschouwd. Bijvoorbeeld, als de IoU-waarde hoger is dan 0,5, wordt de detectie als correct beschouwd.

Figuur 10: Berekening IoU (Persson, 2020)

IoU wordt vaak gebruikt om de nauwkeurigheid van objectdetectie-algoritmen te evalueren en te vergelijken. Door gebruik te maken van IoU als een evaluatiemetriek, kunnen onderzoekers en ontwikkelaars verschillende algoritmen en modellen vergelijken en bepalen welke het beste presteren op hun taken.

3.8 Edge detection

Edge detection of randdetectie is een veelgebruikte techniek in objectdetectie om de grenzen of randen van objecten in een afbeelding te identificeren. Het houdt in dat de veranderingen in pixelwaarden in een afbeelding worden geanalyseerd om gebieden van hoog contrast te lokaliseren, die meestal overeenkomen met randen of grenzen tussen objecten (The MathWorks, Inc., n.d.).

Figuur 11: Edge detection input vs output (Sahir, 2019)

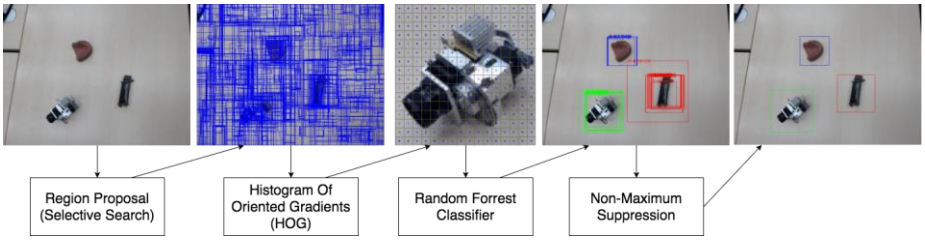
Bij objectdetectie wordt randdetectie vaak gebruikt als een voorbewerkingsstap om kenmerken uit een afbeelding te extraheren die kunnen worden gebruikt om objecten te identificeren. Door de randen van objecten in een afbeelding te identificeren, kan randdetectie helpen bij het onderscheiden van verschillende objecten en belangrijke informatie geven over hun vorm en grootte (Saxena, 2021).

Er zijn verschillende algoritmes voor randdetectie, waaronder de Canny edge detector en de Sobel edge detector, die verschillende wiskundige bewerkingen gebruiken om randen in een afbeelding te identificeren (Haidar, 2021). Zodra de randen zijn geïdentificeerd, kunnen ze verder worden verwerkt om kenmerken zoals hoeken, contouren en textuurpatronen te extraheren, die kunnen worden gebruikt om objecten in een afbeelding te identificeren en te classificeren.

4 OBJECT DETECTIE ALGORITMES

4.1 HOG

In 2005 brachten Dalal en Triggs (2005) een scriptie uit waarin ze Histogram of Oriented Gradients (HOG) voor het eerst aan het licht brachten. HOG wordt gebruikt in computervisie en beeldverwerking met als uiteindelijk doel object detectie. Dit algoritme werd eerder in meerdere computervisie technieken gebruikt, maar is tegenwoordig vervangen voor deep-learning algoritmes zoals CNN's.



Figuur 12: Implementatie HOG (Abouelnaga, 2018)

Het HOG algoritme telt het voorkomen van gradiëntoriëntatie in gelokaliseerde delen van een afbeelding. Het verdeelt de afbeelding in kleine verbonden gebieden die cellen worden genoemd, en voor de pixels in elke cel berekent het HOG algoritme het beeldverloop langs de x-as en de y-as. Deze gradiëntvectoren worden in kaart gebracht met de waardes 0 tot 255. Hierbij zijn pixels met negatieve wijzigingen zwart, pixels met grote positieve wijzigingen zwart en pixels zonder wijzigingen grijs. Met behulp van deze twee waarden wordt het uiteindelijke gradiënt berekend door vectoroptelling uit te voeren. De input (figuur 5) wordt door het HOG algoritme uiteindelijk omgezet naar de output (figuur 6) met een ingewikkelde berekening (Tyagi, 2021). Dit maakt uiteindelijk dat de computer kan zien waar de "B" staat.

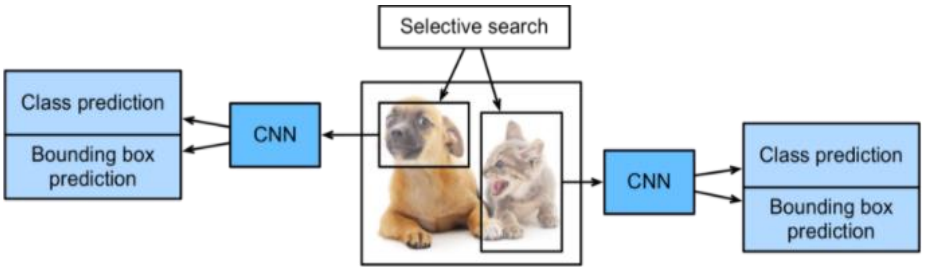
Figuur 13: Input HOG (Tyagi, 2021)

Figuur 14: Output HOG (Tyagi, 2021)

4.2 R-CNN

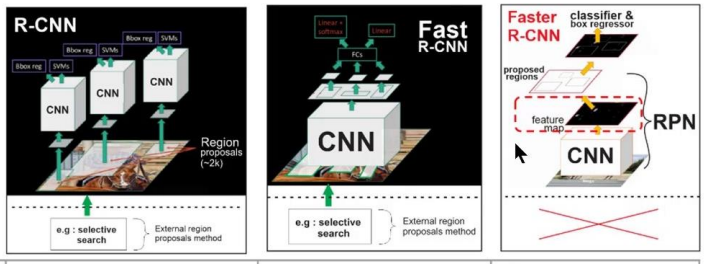
Region-based Convolutional Neural Network (R-CNN) is een populaire deep learning algoritme dat gebruikt wordt voor object detectie. Een Convolution Neural Network (CNN) kan niet omgaan met meerdere instanties van een object of meerdere objecten in een afbeeldingen. Daarom doet een R-CNN eerst aan selectief zoeken binnen een afbeelding, zodat er een voorstel gedaan kan worden waar mogelijk objecten bevinden. Vervolgens wordt elke voorgestelde regio aan een pretrained CNN meegegeven om uiteindelijk met forward propagation[1] te bepalen waar precies een begrenzingskader moet komen te staan. Als het dan mogelijk is kan er door middel van een pretrained SVM gekeken worden wat voor object er binnen het begrenzingskader zit (Singh, 2021).

Met opmerkingen [SdB(3]: Wat betekent SVM?



Figuur 15: Werking R-CNN (Singh, 2021)

[1] Forward propagation is het proces van het doorsturen van invoergegevens door een neuraal netwerk, laag voor laag, om een voorspelling te genereren. Neuronen in elke laag berekenen een lineaire combinatie van de ingangsgegevens met gewichten en biases, gevolgd door een activatiefunctie.



	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

* Standford lecture notes on CNN by Fei Fei Li and Andrej Karpathy 5

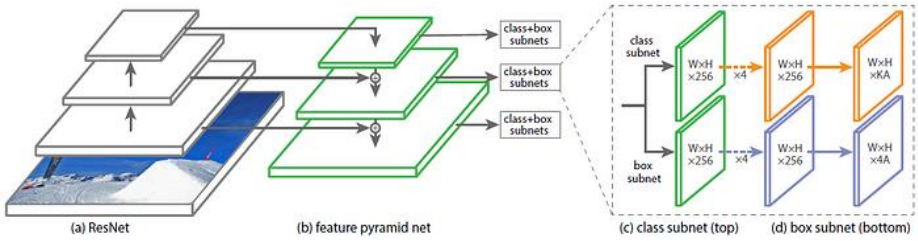
Figuur 16: R-CNN vs Fast R-CNN vs Faster R-CNN (Mu, 2020)

Nu zijn er ook verbeterde versies van R-CNN. Fast R-CNN en Faster R-CNN. Deze technieken scannen de input op andere manieren, waardoor ze sneller werken. In Figuur 16: R-CNN vs Fast R-CNN vs Faster R-CNN is te zien hoe deze technieken werken. Het grote verschil met deze twee technieken is dat er bij Faster R-CNN het selectief zoeken vervangen is met een region proposal network (RPN). Een RPN is een specifiek netwerk dat getrained is om hoge kwaliteit regio voorstellen te doen. Omdat deze hier speciaal voor getrained is, zal een RPN dit accuraat en snel doen (Ren, 2016).

Met opmerkingen [SvE(4]: Hoe werken deze?

4.3 RetinaNET

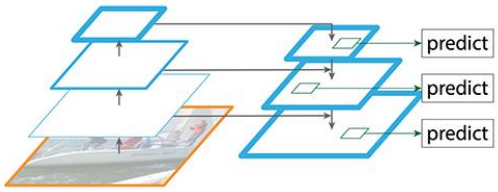
RetinaNet is een populaire algoritme voor objectdetectie dat in 2017 is geïntroduceerd door Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He en Piotr Dollár. Het is ontworpen om het probleem van onevenwichtige datasets in objectdetectie aan te pakken, waarbij objecten van bepaalde klassen veel gebruikelijker zijn dan andere.



Figuur 17: Werking RetinaNet (Tsung-Yi Lin, 2017)

RetinaNet maakt gebruik van een feature pyramid network om een set van feature maps op verschillende ruimtelijke resoluties te genereren uit één enkele invoer afbeelding. Vervolgens past het een nieuw type verliesfunctie, de zogenaamde 'focal loss', toe om een convolutioneel neuraal netwerk (CNN) voor objectdetectie te trainen.

Het feature pyramid network (FPN) is een belangrijk onderdeel van RetinaNet en wordt gebruikt om feature maps op verschillende resoluties te genereren uit een enkele inputafbeelding. Het doel van het FPN is om zowel de ruimtelijke nauwkeurigheid als de semantische rijkheid van de feature maps te verbeteren.



Figuur 18: Feature pyramid network (Hui, 2018)

Het FPN werkt door de feature maps van verschillende lagen in een conventioneel convolutioneel neuraal netwerk (CNN) te combineren en te integreren. Het proces begint bij de onderste laag van het CNN en verloopt dan omhoog door de hiërarchie van feature maps.

In het FPN worden de feature maps op de hogere niveaus van het CNN geïntegreerd met die van de lagere niveaus. Dit gebeurt door middel van up-sampling en additionele convoluties. De resulterende feature maps hebben zowel een hoge ruimtelijke resolutie als een rijke semantische inhoud, waardoor objecten van verschillende groottes en vormen beter kunnen worden gedetecteerd.

Het FPN is vooral nuttig voor objectdetectie omdat objecten in een afbeelding van verschillende groottes kunnen zijn en zich op verschillende locaties kunnen bevinden. Door feature maps op verschillende resoluties te genereren, kan het FPN deze verschillen in grootte en locatie aanpakken en betere detectieresultaten produceren. Dit is vooral belangrijk voor kleine objecten of objecten op afstand, die anders moeilijk te detecteren zouden zijn.

De 'focal loss' vermindert de bijdrage van goed geklasseerde voorbeelden, waardoor de focus van de training wordt gelegd op moeilijke voorbeelden die verkeerd zijn geclassificeerd. Dit helpt om de impact van de onevenwichtigheid in de trainingsdata te verminderen en maakt het mogelijk voor RetinaNet om state-of-the-art prestaties te behalen op objectdetectietaken.

RetinaNet staat bekend om zijn snelheid en nauwkeurigheid en wordt veel gebruikt in verschillende toepassingen zoals autonoom rijden, robotica en beeld- en video-analyse. In Figuur 19: Nauwkeurigheid RetinaNet Is te zien hoe RetinaNet zich qua nauwkeurigheid verhoud tegenover Faster R-CNN, YOLOv2, SSD513 en DSSD513. Hierin is te zien dat RetinaNet de hoogste gemiddelde nauwkeurigheid heeft behaald in de desbetreffende test. In dit figuur staat AP voor Average Precision (gemiddelde nauwkeurigheid) en AP_{50} en AP_{75} voor een Average Precision waarbij de IoU minimaal 0.5 is bij AP_{50} en minimaal 0.75 bij AP_{75}.

	backbone	AP	AP_{50}	AP_{75}
Two-stage methods				
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2
One-stage methods				
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3
RetinaNet (ours)	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**

Figuur 19: Nauwkeurigheid RetinaNet (Boruah, 2021)

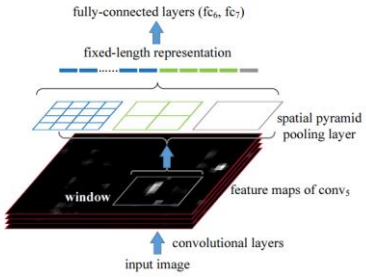
4.4 SPP-net

Met opmerkingen [SvE(5]: Kijk nog even naar https://medium.com/augmented-startups/top-6-object-detection-algorithms-b8e5c41b952f. Hier staat nog een uitleg met een plaatje.

Met opmerkingen [SvE(6R5]: Verder uit vergroten/ scherper krijgen

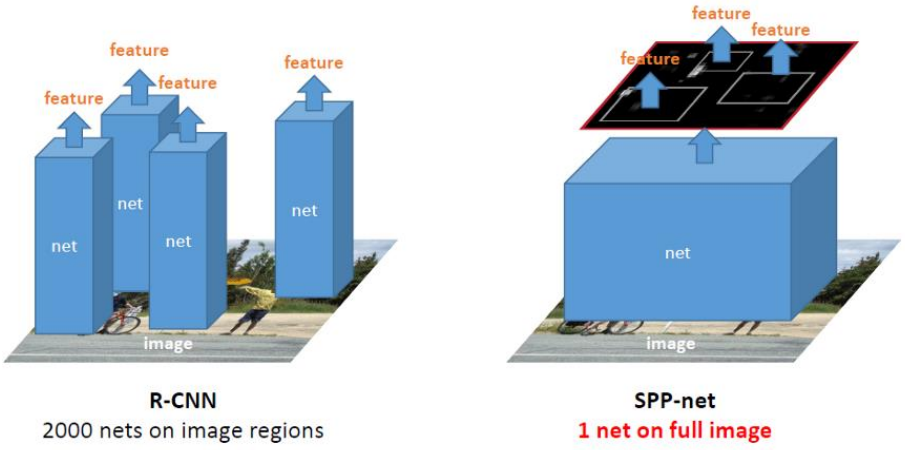
SPP-Net, oftewel Spatial Pyramid Pooling Netwerk, is een deep-learning architectuur die in 2014 werd geïntroduceerd door Kaiming He, Xaingyu Zhang, Shaoqing Ren en Jian Sun van Microsoft Research.

De SPP-Net architectuur is gebaseerd op een CNN en maakt gebruik van een spatial pyramid pooling layer om inputs van verschillende groottes te verwerken. De pooling layer verdeelt de input afbeelding in meerdere sub-regio's en berekent voor elke regio een vaste representatie, ongeacht de grootte. Hierdoor kan het netwerk afbeeldingen van verschillende groottes verwerken zonder dat ze hoeven te worden verkleind of bijgesneden.



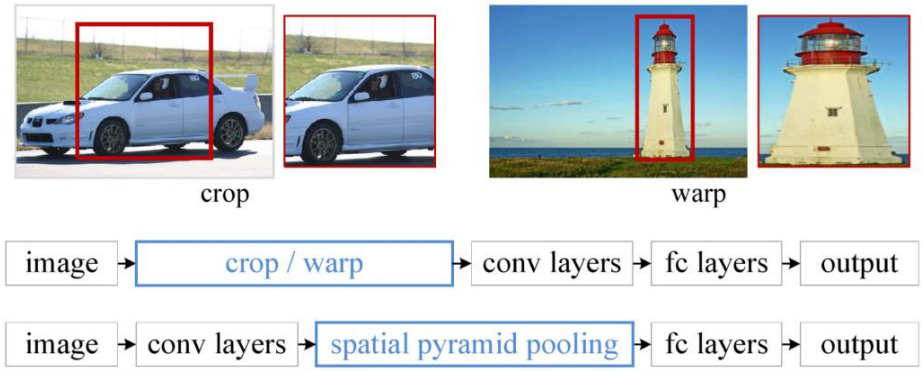
Figuur 20: Fully Connected Layers (He, 2015)

De auteurs van SPP-Net hebben het SPP-mechanisme gebruikt voor object detectie in een verbeterde aanpak. In plaats van de 2000 region proposals één voor één naar het CNN-model te sturen, projecteerden ze de regio's op de Feature-map die verkregen werd uit de 5e convolutional layer. Dit elimineert de 2000 keer dat het CNN-model voor elke afbeelding moet worden doorlopen. Nu gebeurt dit nog maar 1 keer. Echter blijft Selective Search een bottleneck omdat het 2000 proposals moet genereren. Deze regio's worden doorgestuurd naar de SPP-laag voor de pooling in het 1-dimensionale vector. Dit vermindert de rekentijd aanzienlijk. De tijd die nodig is voor een testbeeldvoorspelling op een CPU was binnen 1 seconde en was aanzienlijk sneller in vergelijking met R-CNN en even nauwkeurig.



Figuur 21: R-CNN vs SPP-net (Raikote, n.d.)

Op de PASCAL VOC 2007 dataset behaalde SPP-Net een nauwkeurigheid van ~59% wat hoger was dan de ~54% van R-CNN. En op de ImageNet dataset kon SPP-Net een nauwkeurigheid van ~35% behalen in vergelijken met ~31% voor R-CNN.



Figuur 22: Crop/wrap (He, 2015)

Een voorbeeld van hoe het SPP-Net algoritme kan worden gebruikt, is voor het herkennen van voetgangers in beelden van bewakingscamera's. Dit is een uitdagende taak omdat voetgangers in verschillende houdingen kunnen worden afgebeeld en de beelden vaak van lage kwaliteit zijn. Door het SPP-Net algoritme te trainen op een grote dataset van voetgangersbeelden, kan het nauwkeurig voetgangers herkennen, zelf in beelden van lage kwaliteit.

Tekstueel nog bronverwijzing
https://medium.com/analytics-vidhya/review-spatial-pyramid-pooling-1406-4729-bfc142988dd2
https://appliedsingularity.com/2021/06/01/spatial-pyramid-pooling-in-deep-convolution-networks-sppnet/

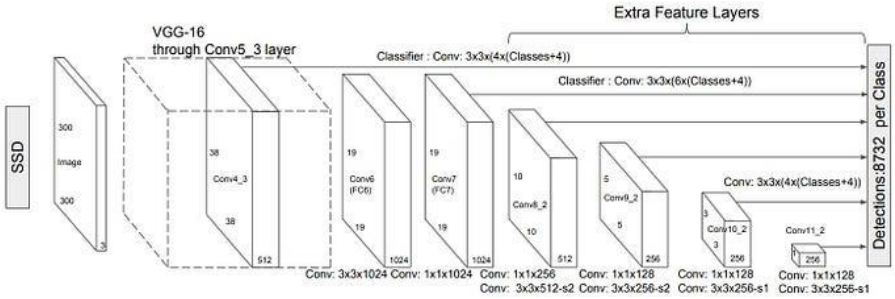
4.5 SSD

Single Shot MultiBox Detector (SSD) is een deep-learning object detective algoritme dat in 2016 werd geïntroduceerd door Wei Liu, Dragomir Anguelov en andere onderzoekers van Google Research.

SSD wordt veel al gebruikt voor het identificeren en lokaliseren van objecten in afbeeldingen of video's. Door de snelheid en nauwkeurigheid waar mee SDD objecten kan vinden is het een populaire keuze voor real-time toepassingen zoals zelfrijdende auto's en videosurveillance-systemen.

SSD werkt door gebruikt te maken van een convolutioneel neuraal netwerk (CNN) om een invoerafbeelding te analyseren en een set begrenzingsvakken (bounding boxes) rond objecten van interesse (ROIs) te definiëren die het algoritme vervolgens in meer detail zal analyseren.

Het SDD-algoritme bestaat uit drie hoofdcomponenten: een basisnetwerk, een set convolutionele feature-maps en een set detectielagen. Het basisnetwerk is meestal vooraf getraind CNN zoals VGG of ResNet, dat wordt gebruikt om functies functies uit de invoerafbeelding te halen.



Figuur 23: Detection using the convolutional predictors (T, n.d.)

De convolutionele feature-maps worden gegenereerd door een set convolutionele filters toe te passen op de uitvoer van het basisnetwerk. Deze feature-maps worden gebruikt om objecten van interesse in de invoerafbeelding te identificeren.

De detectielagen zijn verantwoordelijk voor het generen van de uiteindelijke set begrenzingsvakken rond de gedetecteerde objecten. De detectielagen gebruiken de convolutionele feature-maps om de locatie, grootte en klasse van elk object in de afbeelding te voorspellen.

Een van de belangrijkste kenmerken van SSD is dat het meerdere begrenzingsvakken van verschillende groottes en aspectverhoudingen kan voorspellen voor elk object. Dit maakt het algoritme nauwkeuriger in het detecteren van objecten van verschillende groottes en oriëntaties.

Tekstueel nog bronverwijzing

https://iq.opengenus.org/single-shot-detector/

https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06

4.6 YOLO

Met opmerkingen [SvE(7]: Vergelijking van versies is hier mogelijk handig. (misschien zelfs met plaatje)

Het artikel getiteld "You Only Look Once: Unified, Real-Time Object Detection" werd gepubliceerd in 2016 in de Conference on Computer Vision and Pattern Recognition (CVPR) en ontving de OpenCV People's Choice Award. Het artikel presenteerde een nieuwe benadering voor objectdetectie in realtime met behulp van een volledig convolutioneel neuraal netwerk (FCNN) architectuur genaamd YOLO (You Only Look Once). Deze architectuur gebruikt een enkel netwerk om objectdetectie en classificatie uit te voeren, waardoor het sneller en nauwkeuriger is dan andere op regiovoorstellen gebaseerde technieken zoals Fast R-CNN.

De YOLO-architectuur verdeelt een invoerafbeelding in een S × S raster en elke rastervak is verantwoordelijk voor het detecteren van een object als het midden binnen het vak valt. Elk rastervak voorspelt B afbakeningsvakken en vertrouwensscores voor die vakken, waarbij de vertrouwensscores weergeven hoe zeker het model is dat het vak een object bevat en hoe nauwkeurig het denkt dat het vak is dat het voorspelt. Elk afbakeningsvak bestaat uit 5 voorspellingen: x, y, w, h en vertrouwen, waarbij (x, y) de coördinaten van het midden van het vak zijn ten opzichte van de grenzen van het rastervak, w en h de breedte en hoogte van het voorspelde vak ten opzichte van de hele afbeelding zijn, en de vertrouwensvoorspelling de intersection over union (IOU) vertegenwoordigt tussen het voorspelde vak en elk grondwaarheid vak.

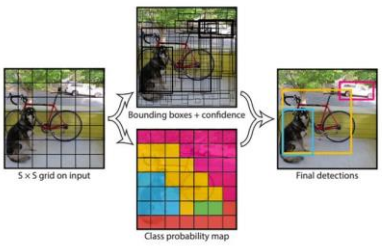


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Het netwerk voorspelt ook C conditionele klassenwaarschijnlijkheden, Pr (Classi | Object), geconditioneerd op het rastervak dat een object bevat, en we voorspellen slechts één set klassenwaarschijnlijkheden per rastervak, ongeacht het aantal vakken B. Bij testtijd worden de conditionele klassenwaarschijnlijkheden en de individuele vertrouwensvoorspellingen van het vak vermenigvuldigd om klasse-specifieke vertrouwensscores voor elk vak te verkrijgen. Deze scores coderen zowel de waarschijnlijkheid van die klasse die in het vak verschijnt als hoe goed het voorspelde vak bij het object past.

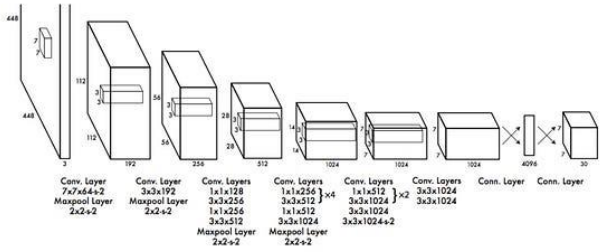
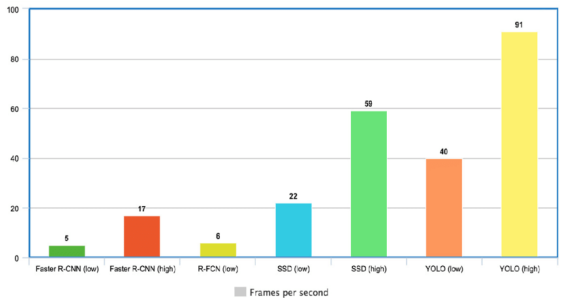


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

De YOLO-architectuur heeft verschillende voordelen ten opzichte van traditionele methoden van objectdetectie. Ten eerste is het extreem snel en kan het streaming video in realtime verwerken met minder dan 25 milliseconden latentie. Ten tweede redeneert YOLO globaal over de afbeelding bij het maken van voorspellingen, in tegenstelling tot sliding window- en region proposal-gebaseerde technieken die slechts een deel van de afbeelding tegelijk zien. Dit globale redeneren stelt YOLO in staat om contextuele informatie over klassen en hun uiterlijk impliciet te coderen, waardoor het minder foutgevoelig is bij achtergrondpatches. Ten derde leert YOLO generaliseerbare representaties van objecten, waardoor het minder waarschijnlijk is om te falen bij nieuwe domeinen of onverwachte invoer.



Figuur 24: FPS vergelijking Faster R-CNN, SSD en YOLO (Hernández, 2020)

YOLO staat bekend om zijn ondersteuning van hoge frames per second (FPS). Dit maakt voor afbeeldingen niet uit, maar is voor video's wel erg belangrijk. Een video bestaat uit meerdere frames die snel achter elkaar worden afgespeeld. Faster R-CNN en SSD ondersteunen een lager aantal frames per second, waar Faster R-CNN aanzienlijk lager is. In *Figuur 24: FPS* vergelijking Faster R-CNN, SSD en YOLO is deze vergelijking te zien, maar moet wel rekening gehouden worden met dat er een vergelijking is gemaakt met YOLO v2. Tegenwoordig bestaat ook al YOLO v8, welke nog sneller is.

Het YOLOv2-paper, gepubliceerd in 2016, introduceerde verschillende verbeteringen ten opzichte van de originele YOLO-architectuur. Een van de belangrijkste verbeteringen was het gebruik van batchnormalisatie, waardoor overfitting werd verminderd en het netwerk robuuster werd. YOLOv2 gebruikte ook anchor-boxes om de lokaliseringsnauwkeurigheid te verbeteren en gebruikte een nieuwe loss-functie die fouten in kleine objecten meer bestrafte dan fouten in grote objecten. YOLOv2 was sneller en nauwkeuriger dan de oorspronkelijke YOLO-architectuur en behaalde state-of-the-art prestaties op verschillende objectdetectiedatasets.

Concluderend zijn de YOLO-architectuur gepresenteerd in het oorspronkelijke paper en de verbeterde versie, YOLOv2, beide zeer efficiënte en nauwkeurige methoden voor real-time objectdetectie. De YOLO-architectuur is uniek omdat het een enkel netwerk gebruikt om zowel objectdetectie als classificatie uit te voeren, wat het sneller en nauwkeuriger maakt dan andere region proposal-gebaseerde technieken. De mogelijkheid van de architectuur om globaal te redeneren over de afbeelding bij het maken van voorspellingen en om generaliseerbare representaties van objecten te leren, maakt het zeer effectief in een verscheidenheid van domeinen en toepassingen.

Tekstueel nog bronverwijzing

https://arxiv.org/abs/1506.02640
https://viso.ai/deep-learning/object-detection/#:~:text=Popular%20algorithms%20used%20to%20perform,the%20single%2Dshot%20detector%20family.
https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

https://medium.com/augmented-startups/top-6-object-detection-algorithms-b8e5c41b952f

5 CONCLUSIE

Object detectie is een technologie binnen het vakgebied van computervisie en kunstmatige intelligentie, waarbij computersystemen worden getraind om objecten in afbeeldingen of videobeelden te herkennen en lokaliseren. Het doel van object detectie is om een computersysteem te ontwikkelen dat automatisch objecten kan detecteren in beelden zonder menselijke tussenkomst.

Op basis van de beantwoording van de deelvragen, blijkt dat object detectie een belangrijke technologie is die in diverse toepassingen wordt gebruikt. Er zijn verschillende technieken die momenteel worden gebruikt voor object detectie, waarbij de voor- en nadelen per techniek verschillen. Bij het vergelijken van de verschillende algoritmes voor object detectie, is gebleken dat SSD, YOLO en faster R-CNN veelbelovende algoritmes zijn met hoge nauwkeurigheid en snelheid.

Bij het kiezen van het meest geschikte algoritme voor object detectie moet er ook rekening gehouden worden met andere factoren, zoals de toepassing en het gebruiksscenario. Zo kan bijvoorbeeld de snellere versie van R-CNN, faster R-CNN, een goede optie zijn voor het detecteren van objecten in stilstaande beelden. Echter, als er ook video's geanalyseerd moeten worden, dan is YOLO een betere optie vanwege de hogere frames per seconde (FPS) die worden ondersteund. Daarom is het belangrijk om bij het kiezen van een algoritme rekening te houden met de specifieke eisen van de toepassing.

Het is interessant om een vervolgonderzoek te doen naar de werking van de YOLO en SSD algoritmes door middel van prototyping. Door middel van het ontwikkelen van prototypes kan meer inzicht worden verkregen in de werking van de algoritmes en kan de nauwkeurigheid verder worden bestudeerd.

LITERATUURLIJST

The MathWorks, Inc. (sd). *Why Object Detection Matters*. Opgehaald van MathWorks: https://nl.mathworks.com/discovery/object-detection.html

Abouelnaga, Y. (2018, 01 29). *Object Detection: Histogram Of Gradients and Random Forest*. Opgehaald van Abouelnaga: https://abouelnaga.io/projects/hog-random-forest-object-detection/

Boruah, P. (2021, 05 12). *RetinaNet: The beauty of Focal Loss*. Opgehaald van TowardsDataScience: https://towardsdatascience.com/retinanet-the-beauty-of-focal-loss-e9ab132f2981

Dalal, N. (2005). *Histograms of Oriented Gradients for Human Detection*. Opgehaald van https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

DeepLearningAI. (2017, 11 7). Opgehaald van Youtube: https://www.youtube.com/watch?v=RTlwl2bv0Tg

Duursma, J. (sd). *Wat is een neuraal netwerk?* Opgehaald van Jarno: https://jarnoduursma.nl/blog/is-neuraal-netwerk/

Goglia, D. (2021, 07 27). *Backpropagation for Dummies*. Opgehaald van Medium: https://medium.com/analytics-vidhya/backpropagation-for-dummies-e069410fa585

Haidar, L. (2021, 09 11). *Sobel vs. Canny Edge Detection Techniques*. Opgehaald van Medium: https://medium.com/@haidarlina4/sobel-vs-canny-edge-detection-techniques-step-by-step-implementation-11ae6103a56a#:~:text=The%20main%20advantages%20of%20the,Non%2Dmaxima%20suppression%20and%20thresholding.

He, K. (2015, 04 23). *Spatial Pyramid Pooling in Deep Convolutional*. Opgehaald van Arxiv: https://arxiv.org/pdf/1406.4729v4.pdf

Hernández, S. A. (2020, 06). *A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework*. Opgehaald van ResearchGate: https://www.researchgate.net/publication/342570032_A_review_Comparison_of_performance_metrics_of_pretrained_models_for_object_detection_using_the_TensorFlow_framework

Hui, J. (2018, 03 27). *Understanding Feature Pyramid Networks for object detection*. Opgehaald van Medium: https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

Loy, J. (2019). *Neural Network Projects With Python.* Packt Publishing. Opgehaald van https://www.oreilly.com/library/view/neural-network-projects/9781789138900/e1f93bb9-0e51-428d-8e06-f19143ecc927.xhtml

Mu, M. (2020, 03 23). *Product detection in movies and TV shows using machine learning – part 1: background*. Opgehaald van Dr Mu Mu: https://drmu.net/2020/03/23/product-detection-in-movies-and-tv-shows-using-machine-learning-part-1-background/

Persson, A. (2020, 10 5). *Intersection over Union Explained and PyTorch Implementation*. Opgehaald van Youtube: https://www.youtube.com/watch?v=XXYG5ZWtjj0

Raikote, P. (sd). Opgehaald van Applied Singularity: https://appliedsingularity.com/2021/06/01/spatial-pyramid-pooling-in-deep-convolution-networks-sppnet/

Ren, S. (2016, 01 06). *Faster R-CNN: Towards Real-Time Object*. Opgehaald van Arxiv: https://arxiv.org/pdf/1506.01497v3.pdf

Rosebrock, A. (2015, 03 23). *Sliding Windows for Object Detection with Python and OpenCV*. Opgehaald van Pyimagesearch: https://pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv//

Sahir, S. (2019, 01 25). *Edge Detection Step by Step in Python — Computer Vision*. Opgehaald van Towards Data Science: https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

Saxena, S. (2021, 03 12). *Edge Detection: Extracting The Edges From An Image*. Opgehaald van Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/03/edge-detection-extracting-the-edges-from-an-image/

Singh, A. (2021, 09 03). *Top 6 Object Detection Algorithms*. Opgehaald van Medium: https://medium.com/augmented-startups/top-6-object-detection-algorithms-b8e5c41b952f

Subramanyam, V. (2021, 01 20). *Non Max Suppression (NMS)*. Opgehaald van Medium: https://medium.com/analytics-vidhya/non-max-suppression-nms-6623e6572536

T, A. N. (sd). *Single Shot Detector (SSD) + Architecture of SSD*. Opgehaald van Open Genus: https://iq.opengenus.org/single-shot-detector/

The MathWorks, Inc. (sd). *Anchor Boxes for Object Detection*. Opgehaald van MathWorks: https://www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html

The MathWorks, Inc. (sd). *Mathworks*. Opgehaald van Edge Detection: https://nl.mathworks.com/discovery/edge-detection.html

Tsung-Yi Lin, P. G. (2017, 10 7). *Focal Loss for Dense Object Detection*. Opgehaald van Arxiv: https://arxiv.org/pdf/1708.02002v2.pdf

Tyagi, M. (2021, 07 04). *HOG (Histogram of Oriented Gradients): An Overview*. Opgehaald van Towards Data Science: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f#:~:text=What%20is%20it%3F,the%20purpose%20of%20object%20detection

https://medium.com/analytics-vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81

https://www.baeldung.com/cs/object-recognition-tasks-differences

https://www.oreilly.com/library/view/neural-network-projects/9781789138900/e1f93bb9-0e51-428d-8e06-f19143ecc927.xhtml

https://towardsdatascience.com/object-localization-in-overfeat-5bb2f7328b62#:~:text=The%20difference%20between%20object%20localization,the%20objects%20and%20their%20boundaries.

https://arxiv.org/pdf/1512.03385v1.pdf

https://www.fritz.ai/object-detection/#:~:text=Object%20detection%20is%20a%20computer,all%20while%20accurately%20labeling%20them.

https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

https://www.einfochips.com/blog/understanding-object-localization-with-deep-learning/

https://towardsdatascience.com/localization-and-object-detection-with-deep-learning-67b5aca67f22

https://arxiv.org/pdf/1311.2524.pdf

OPEN UP
NEW HAN_UNIVERSITY OF APPLIED SCIENCES
HORIZONS.