Objectdetectie van motoren

Eindverlag Vision

Roel Stierum

A motorcycle parked in a driveway

Description automatically generated

# Introductie

Voor de opdracht moest ik kiezen wat voor object ik wilde detecteren. Mijn keuze viel op het detecteren van motoren, het motorrijden is een erg grote hobby van mij. Het doel was om een computermodel te trainen dat in staat is om accuraat en efficiënt motoren te herkennen op afbeeldingen. Dit moet gedaan worden met behulp van een dataset met afbeeldingen waarin motoren waren aangeduid

Het hele proces bestond uit een aantal belangrijke stappen. Eerst moest ik een geschikte dataset selecteren en voorbereiden. Dit betekende dat ik afbeeldingen moest verzamelen waarin motoren te zien waren en deze moest annoteren, zodat het model kon leren om ze te herkennen. Daarna heb ik een neuraal netwerk getraind met behulp van een aangepast model, zodat het specifiek gericht was op het vinden van motoren op afbeeldingen.

Een belangrijk onderdeel van mijn onderzoek was het testen van het getrainde model. Hiervoor heb ik een experiment opgezet om te zien hoe nauwkeurig het model motoren kon vinden in afbeeldingen. Dit deed ik met behulp van een techniek genaamd Intersection over Union (IoU), die de mate van overlap tussen de voorspelde en echte locaties van motoren meet.

Dit rapport beschrijft niet alleen stap voor stap hoe ik te werk ben gegaan bij het voorbereiden van de data en het trainen van het model, maar laat ook zien hoe het getrainde model presteert en welke uitdagingen ik ben tegengekomen. Het geeft een goed beeld van wat er mogelijk is met dit soort technieken en waar de beperkingen liggen bij het detecteren van motoren op afbeeldingen.

# Achtergrond

In dit hoofdstuk wordt een overzicht gegeven van de relevante literatuur met betrekking tot het detecteren van objecten in afbeeldingen, met specifieke aandacht voor neurale netwerken en deep learning.

## Objectdetectie met Neurale Netwerken

Objectdetectie in afbeeldingen is een cruciale taak binnen het domein van computervisie en heeft de afgelopen jaren aanzienlijke vooruitgang geboekt dankzij de opkomst van neurale netwerken. Traditionele benaderingen voor objectdetectie maakten vaak gebruik van handmatig ontworpen functies en classificatiemethoden, wat beperkingen met zich meebracht in termen van nauwkeurigheid en robuustheid [@ren2015faster].

Neurale netwerken hebben deze benadering getransformeerd door end-to-end leren mogelijk te maken, waarbij het model zowel de functies leert als de classificatie uitvoert. Dit heeft geleid tot verbeterde prestaties in objectdetectie, met name met betrekking tot complexe scenario's en objecten met variabele aspecten [@liu2016ssd].

## Deep learning voor Objectdetectie

Binnen het paradigma van diep leren hebben verschillende architecturen significant bijgedragen aan objectdetectie. Een populaire aanpak is het gebruik van Convolutional Neural Networks (CNN's), die uitstekende prestaties hebben laten zien in het leren van kenmerken van afbeeldingen op verschillende abstractieniveaus. Deze kenmerken worden vervolgens gebruikt voor zowel classificatie als lokalisatie van objecten [@redmon2016you].

Een andere belangrijke ontwikkeling is de introductie van region proposal networks (RPN's) binnen het framework van Faster R-CNN en andere gelijkaardige architecturen. Deze mechanismen maken het mogelijk om efficiënt potentiële regio's van interesse te genereren, waardoor de detectieprestaties verder worden verbeterd [@ren2015faster].

## Toepassing op Specifieke Objecten

Hoewel objectdetectie in het algemeen aanzienlijke vooruitgang heeft geboekt, blijven er uitdagingen bestaan bij het detecteren van specifieke objecten, zoals motoren. Deze objecten kunnen variëren in grootte, vorm en oriëntatie, wat de detectie bemoeilijkt, vooral in complexe omgevingen.

Recente studies hebben zich gericht op het verbeteren van de nauwkeurigheid en robuustheid van objectdetectiealgoritmen, met name door middel van geavanceerde architecturale aanpassingen en verfijnde trainingsmethoden. Het begrip van deze ontwikkelingen is essentieel voor het ontwerpen en trainen van effectieve modellen voor het detecteren van motoren en andere objecten in afbeeldingen [@bochkovskiy2020yolov4].

## Conclusie

Dit hoofdstuk heeft een overzicht gegeven van de relevante literatuur met betrekking tot objectdetectie in afbeeldingen, met een focus op de toepassing van neurale netwerken en deep learning. Het biedt een context voor het begrijpen van de benaderingen en technieken die worden gebruikt in het ontwerp en de training van modellen voor het detecteren van objecten, zoals motoren, binnen computervisie.

# Methodebeschrijving

Om het probleem van het detecteren van motoren in afbeeldingen aan te pakken, heb ik een systematische aanpak gevolgd, bestaande uit verschillende stappen:

## 1. Dataset Laden en Verwerken

Eerst heb ik de dataset geladen en verwerkt om de afbeeldingen en bijbehorende bounding box-coördinaten te verkrijgen. De loadDataSet-functie laadt afbeeldingen en bijbehorende XML-bestanden die bounding box-informatie bevatten. De afbeeldingen worden omgezet naar het verwachte formaat van het model (224x224x3) en genormaliseerd.

# Code snippet: loadDataSet function

def loadDataSet(dataset\_path='dataset', img\_size=(224, 224)):

    # Function definition here...

    return (data\_dict['train']['x'], data\_dict['train']['y'], data\_dict['test']['x'], data\_dict['test']['y'])

X\_train, Y\_train, X\_test, Y\_test = loadDataSet()

figuur 1 dataset inladen

## 2. Model Aanpassen

Om het VGG16-model aan te passen voor objectdetectie, heb ik de volgende stappen uitgevoerd:

1. **Topclassificatielagen Verwijderen en Aanpassen:** Ik heb de topclassificatielagen van het VGG16-model verwijderd, aangezien deze waren geoptimaliseerd voor beeldclassificatie en niet geschikt waren voor objectdetectie. Vervolgens heb ik aangepaste lagen toegevoegd voor objectdetectie.
2. **Transfer Learning Toepassen:** Voor het aanpassen van het VGG16-model heb ik gebruikgemaakt van transfer learning. Hierbij heb ik het VGG16-model geladen met de gewichten van 'imagenet' als startpunt. Dit maakt het mogelijk om te profiteren van de reeds geleerde kenmerken van het model.
3. **Toevoegen van Flatten Layer:** Als eerste aanpassing heb ik een Flatten-laag toegevoegd aan het model. Deze laag is cruciaal om de uitvoer van het convolutionele deel om te zetten naar een eendimensionale tensor. Hierdoor kan de uitvoer worden gebruikt als input voor de volgende dense laag.
4. **Toevoegen van Aangepaste Lagen voor Objectdetectie:** Na de Flatten-laag heb ik aangepaste lagen toegevoegd voor objectdetectie. Dit omvatte het toevoegen van een Dense-laag met 256 units en ReLU-activatiefunctie voor het uitvoeren van classificatie op basis van de geleerde kenmerken. Vervolgens werd een uitvoerlaag toegevoegd met 4 units en een lineaire activatiefunctie voor het voorspellen van de bounding box-coördinaten van het beoogde object in de afbeelding.

Deze stappen hebben het VGG16-model succesvol aangepast voor objectdetectie, waardoor het model in staat is om nauwkeurig de bounding box-coördinaten van motoren in afbeeldingen te voorspellen.

# Code snippet: Model aanpassen

# Load VGG16 model without the top classification layer

base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=input\_shape)

# Add a flatten layer and a dense layer for classification

x = Flatten()(base\_model.output)

x = Dense(256, activation='relu')(x)

output = Dense(4, activation='linear')(x)

# Create a new model with input and output layers

model = Model(inputs=base\_model.input, outputs=output)

# Freeze the weights of the pre-trained layers

for layer in base\_model.layers:

    layer.trainable = False

# Compile the model

model.compile(loss='mean\_squared\_error', optimizer='adam', metrics=['mean\_squared\_error'])

figuur 2 model aanpassen

## 3. Model Training

Het aangepaste model is vervolgens getraind op de aangepaste dataset met behulp van de Adam-optimizer. Tijdens het trainen heb ik de prestaties geëvalueerd met behulp van de Mean Squared Error (MSE).

# Code snippet: Model Training

hist = model.fit(X\_train, Y\_train, epochs=12, validation\_data=(X\_test, Y\_test), batch\_size=64)

figuur 3 model trainen

## 4. Evaluatie van het Getrainde Model

Na het trainen van het model heb ik de prestaties geëvalueerd door de Intersection over Union (IoU) te berekenen tussen de voorspelde bounding boxes en de ware bounding boxes op de testdataset. Hierbij heb ik ook de gemiddelde IoU-score berekend voor alle afbeeldingen in de testdataset.

# Code snippet: Evaluatie van het Getrainde Model

score = 0

for index, item in enumerate(X\_test):

    image = np.expand\_dims(X\_test[index], axis=0)

    Pred\_Bbox\_Label = tuple(map(int, model.predict(image)[0]))

    iouScore = (calculate\_IoU(Pred\_Bbox\_Label, Y\_test[index]))

    score+=(iouScore)

    display\_image(X\_test[index], [Y\_test[index], Pred\_Bbox\_Label])

print(score/len(X\_test))

figuur 4 model evalueren

Deze stappen hebben geresulteerd in een getraind model dat in staat is om redelijk nauwkeurig motoren te detecteren binnen afbeeldingen.

# Beschrijving van experiment

Het doel van dit experiment was om de nauwkeurigheid van het getrainde model te evalueren door gebruik te maken van de Intersection over Union (IoU) als evaluatiemethode. IoU is een veelgebruikte maatstaf in objectdetectie om de mate van overlapping tussen voorspelde en werkelijke objectgrenzen te meten.

## Experimentele Opzet:

**Dataset Evaluatie:** Voorafgaand aan de evaluatie heb ik de dataset opgedeeld in trainings- en testsets. De trainingsset werd gebruikt om het model te trainen, terwijl de testset werd gebruikt om de prestaties van het model te evalueren.

**Model Evaluatie:** Voor elke afbeelding in de testset heb ik het getrainde model toegepast om de bounding box-coördinaten van het beoogde object te voorspellen. Vervolgens heb ik de IoU tussen de voorspelde bounding box en de werkelijke bounding box berekend.

**Berekening van de IoU:** De IoU-waarde werd berekend met behulp van de formule: IoU=Intersection AreaUnion AreaIoU=Union AreaIntersection Area​ Hierbij werd de Intersection Area berekend als het overlappende gebied tussen de voorspelde en werkelijke bounding boxes, terwijl de Union Area de totale ruimte bedekt door beide bounding boxes vertegenwoordigde.

**Resultaten Analyse:** Na het berekenen van de IoU-waarden voor alle afbeeldingen in de testset, werden de gemiddelde IoU-waarden berekend om de algehele nauwkeurigheid van het model te beoordelen. Daarnaast werden visualisaties gebruikt om de voorspelde bounding boxes te vergelijken met de werkelijke bounding boxes om eventuele fouten of onnauwkeurigheden te identificeren.

#### Resultaten:

Het experiment resulteerde in een gemiddelde IoU-score van 77,05% die de mate van nauwkeurigheid van het model weergaf bij het voorspellen van de bounding box-coördinaten van motoren in afbeeldingen. De visualisaties van de voorspelde en werkelijke bounding boxes werden gebruikt om de prestaties van het model te analyseren.

A close-up of a motorcycle

Description automatically generated

figuur 5 Objectdetectie onder optimale omstandigheden

De afbeelding uit figuur 5 heeft een IoU score van 97,18%. Dat is zeer hoog, kennelijk is de orientatie van de motor en het onderscheid tussen motor en achtergrond zeer goed te zien met het getrainde model. Dit komt waarschijnlijk doordat veel van de afbeeldingen in de traindata op deze afbeelding lijken

A two motorcycles on a black background

Description automatically generated

figuur 6 Objectdetectie onder middelmatige omstandigheden

De afbeelding uit figuur 6 heeft een IoU score van 77,94%. Dat is al aanmerkelijk lager dan die uit figuur 5. Dit heeft te maken met dat de motor uit een minder haakse hoek is gefotografeerd dan de meeste motoren uit de dataset.

A couple of people on a motorcycle

Description automatically generated

figuur 7 Objectdetectie onder slechte omstandigheden

De afbeelding uit figuur heeft een score van 40,28%. Dit is de laagste uit de evaluatiedata. Dit komt omdat er personen op de motor zitten en de motor ook nog eens onder een scherpe hoek gefotografeerd is. Er zaten maar weinig vergelijkbare foto’s in de train dataset.

#### Conclusie:

Het experiment gaf inzicht in de nauwkeurigheid en robuustheid van het getrainde model voor het detecteren van motoren in afbeeldingen. Een gemiddelde IoU-score van 77,05% is best goed. Het model lijkt het meeste last te hebben bij het detecteren van motoren waar de achtergrond niet enorm veel contrast heeft met de motor en als er personen op de motor zitten.

# Referenties

* Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).
* Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.
* Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
* Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934.