

**COMPUTER**

**VISION**

**PLANNI**

**NG**

***Bo***

***otbreuk***

***detectie***

Inhoud

[INLEIDING 2](#_Toc201703289)

[1. DEFINITIE VAN METINGEN 3](#_Toc201703290)

[2. Wijzigingen 5](#_Toc201703291)

[3. Fases 5](#_Toc201703292)

[3.1 Dataset manipulatie 5](#_Toc201703293)

[3.2 Analyse van het einddoel 5](#_Toc201703294)

[3.3 Training 6](#_Toc201703295)

# INLEIDING

In dit document wordt het stappenplan uitgebreid beschreven dat nodig is om de Computer Vision-challenge succesvol af te ronden. Hier vindt de lezer de methoden die worden geïmplementeerd om het doel te kunnen bereiken. Zoals eerder in de challenge-keuze miniverslag werd aangegeven, deze uitdaging heeft als einddoel het detecteren van bootbreuken, deze te markeren met een rode vierkante en tot een conclusie te komen.

Metrieken en data wordt ook aangegeven die nodig zijn. Hiervoor wordt er ook een apart test set gezet.

Het is belangrijk dat de methoden goed worden vastgesteld op basis van een taak analyse. Hier denken we aan welke kenmerken de taak heeft of in andere woorden welke eigenschappen de doelen hebben.

# DEFINITIE VAN METINGEN

De keuze van metingen en het type data dat nodig is, hangt af van de specifieke kenmerken van het doel. Aangezien het doel de detectie van kleine scheurtjes op boten betreft, waarbij deze scheurtjes **verschillende vormen kunnen** hebben, is het essentieel dat de dataset voldoende groot is. Bovendien is het ook van belang dat er ook rekening moet worden gehouden met **de willekeurige positie van een scheurtjes** op het afbeelding. De volgende punten geven verdere belangrijke eigenschappen van het doel aan:

* **Varietëit aan texturen:** In de gebruikte dataset zijn er texturen die op bot lijken zoals spiervezels en gipshoezen.
* **Omgevingsfactoren:** In sommige gevallen kunnen licht en reflectie van vloeistof de detectie op een ongewenste manier beïnvloeden.

Op basis van de bovengenoemde factoren en analyse kunnen we vaststellen welke metingen van belang zijn. Hieronder worden ze aangegeven:

* **Nauwkeurigheid**: Het percentage correcte classificaties van de methode. Hier moeten we met het volgende punten rekening houden. Dedataset moet goed gebalanceerd zijn, het aantal TP plaatjes moet ongeveer hetzelfde zijn als die van TN plaatjes. Dit voorkomt een bias-gedrag bij het voorspellen van resultaten. Nauwkeurigheid ga ik met het volgende formule berekenen:

Nauwkeurigheid = 𝐴𝑎𝑛𝑡𝑎𝑙 𝑐𝑜𝑟𝑟𝑒𝑐𝑡𝑒 𝑣𝑜𝑜𝑟𝑠𝑝𝑒𝑙𝑙𝑖𝑛𝑔𝑒𝑛 𝑥 100%

𝑇𝑜𝑡𝑎𝑙𝑒 𝑎𝑎𝑛𝑡𝑎𝑙 𝑣𝑜𝑜𝑟𝑠𝑝𝑒𝑙𝑙𝑖𝑛𝑔𝑒𝑛

* **Confusion Matrix**: Dit geeft een overzicht weer van de resultaten waar true positives(TP), true negatives (TN), false positives (FP) en false negatives (FN) worden ingedeeld.

1. True Positive: In het plaatje komt een bootbreuk voor en het model geef een 0 als er wel een botbreuk in zit.
2. False Positive: In het plaatje wordt iets vergelijkbaars met een bootbreuk en de botbreuk geeft een 0 aan.
3. True Negative: Er zit geen bootbreuk in het plaatje en er wordt een 1 als output verwacht
4. False Negative: Er zit geen bootbreuk in het plaatje maar het model geeft aan dat er een breuk in zit.

De reden waarom er een 0 wordt verwacht komt doordat tensorflow de custom dataset laad op basis van de subdirectory positie. Bij de aangemaakte subsets is de volgorde altijd “fracture” en “no\_fracture”. Ik heb het zo willen laten omdat het makkelijker voor mij is in verband met de naamgeving van de aangemaakte mappen.

# Wijzigingen

In principe wilde ik dit project inrichten naar het detecteren van algemene bootbreuk, maar ik kwam erachter dat de dataset bestemd was voor YOLOv8 model. Dit model is getraind en ontwikkeld in Pytorch framework wat voor deze opdracht niet toegestaan is. Hierdoor was ik van plan om alles anders aanpaken.

Nu wordt het Object detection task gericht op de detectie van voorarm breuken in plaats van algemene boten omdat deze de theoretisch gezien eenvoudig is in vergelijking met andere soort breuken zoals in de pols waarbij meerdere boten zijn die de complexiteit van de training kan verhogen en degene die het meest voorkomt in de dataset.

# Fases

## 3.1 Dataset manipulatie

3.1.1 Labels

De dataset in Kaggle heeft een bijzondere manier van labelen. De labels zijn in YOLOv8 formaat, volgens het antwoord van K M Kung in de discussie sectie van het profiel van de dataset. De labels zijn als volg:   
(classLabel, x1, y1, x2, y2, x3, y3, x4, y4, x1, y1)

Vervolgens zijn de label files van gezonde boten leeg.

3.1.2 Images

Ik filtreer de images op basis van de classLabel uit de label bestanden en knip op basis van de aangegeven coördinaten waar de botbreuk zit. Vervolgens sla ik deze images op wat in een custom validation, training en test “subsets” resulteert. Deze worden later ingeladen voor Binary Classification waarbij de model de aanwezigheid van een breuk leert.

## 3.2 Analyse van het einddoel

Voor de detectie van bootbreuken zijn er medische concepten nodig. De structuur van de bot speelt een grote rol en het verschillen van deze met andere texturen is ook van belang. Een voorbeeld hiervan is *soft tissue* van de spieren rond de bot. Deze kunnen het model belemmeren om de juiste edges van de bot en scherpe kanten van de breuk te highlighten. Hiervoor moet er een manier worden uitgevonden om de image te highlighten. Er zijn verschillende contrast en edge detection methodes die kunnen worden overwogen en getest op de image. Hieronder is er een lijst daarvan:

1. CLAHE (*Contrast Limited Adaptive Histogram Equalization*) voor het highighten van patronen door het contrast op basis bepaalde gebied van de image aan te passen en niet op de volledige image.
2. Sobel. Deze kan worden gebruikt voor de detectie van scherpen randen (de breuk zelf)
3. Gabor. Ook een edge detectie methode maar het is complexer in de praktijk en richt zich meer op de textuur.
4. Gaussian Filter.

## 3.3 Training

Voor de initiële training start ik met een model dat een relatief groot aantal parameters bevat. Dit stelt me in staat om te evalueren of de gekozen architectuur in staat is om relevante kenmerken uit de data te extraheren. De prestaties zal ik analyseren op basis van training- en validatieloss. Afhankelijk van deze resultaten zal ik de complexiteit van het modelijk gelidelijk verlagen. Hiervoor kan ik bijvoorbeeld het aantal feature maps verlagen. De toevoegeing van een dense layer is ook voordelig bij abstracte patronen, zoals het concept van welke image een botbreuk is en welke niet.

De training data is evenveel opgesplitst voor beide klasses (aanwezig of niet). Bij de niet aanwezig zal ik images random kiezen uit de training dataset.

## 3.4 Testing

Voor het testen gebruik ik een confusion matrix om te beoordelen of het model een bias heeft richting een van de twee klassen (aanwezigheid vs. afwezigheid van een bot).

De accuracy bereken ik door het model meerdere testafbeeldingen te laten classificeren en het gemiddelde aantal correcte voorspellingen te bepalen.

De nauwkeurigheid evalueer ik op een grotere afbeelding waarbij het model niet alleen moet aangeven of er een bot is, maar ook waar deze zich bevindt. Dit meet hoe precies het model de locatie kan voorspellen.