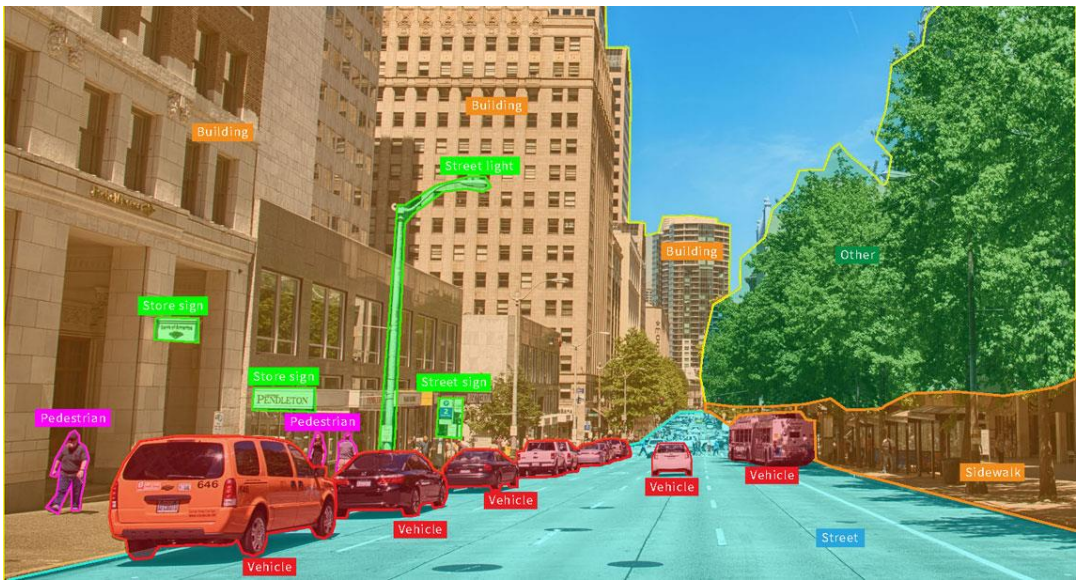
2023



VISN | Eindopdracht

CAR OBJECT DETECTION WITH CUSTOM VGG16-MODEL

Ahmet Serdar Çanak

Datum:	14-12-2023
Cursuscode:	TICT-TV2VISN1-20
Naam:	Ahmet Serdar Çanak
Studierichting:	Technische Informatica
Specialisatie:	Embedded Systems Engineering
Studentennummer:	1760039

Inhoud

1. Introductie

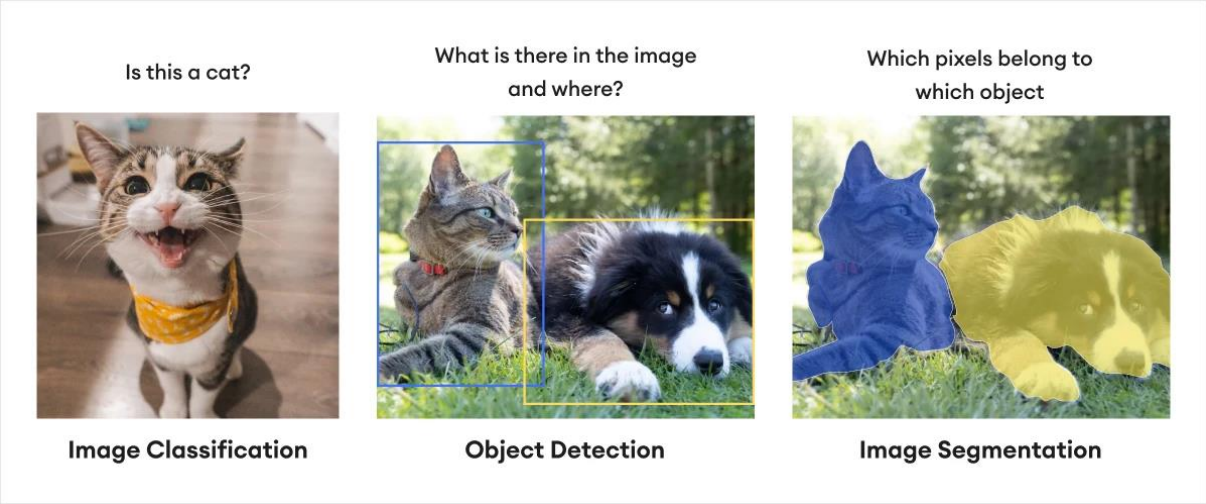
Computervisie heeft de afgelopen jaren aanzienlijke vooruitgang geboekt dankzij de opkomst van geavanceerde technieken zoals neurale netwerken en diepgaand leren. In deze eindrapportage wordt een diepgaande verkenning gepresenteerd van een specifieke uitdaging binnen dit domein, waarbij de focus ligt op Image Classification, Object Detection en Image Segmentation.

De uitdaging omvatte de keuze tussen verschillende taken binnen het domein van computervisie, waarbij ik heb gekozen voor objectdetectie met behulp van een bestaande dataset van geannoteerde afbeeldingen met betrekking tot auto's. Het doel was om een neurale netwerkarchitectuur te trainen die in staat is om nauwkeurig en efficiënt auto's te detecteren binnen afbeeldingen.

Het proces omvatte verschillende cruciale stappen, beginnend met het selecteren en voorbereiden van de dataset. Hierbij werd gebruik gemaakt van beeldverwerkingsmethoden om een voorgefilterde dataset te creëren, waarin relevante informatie met betrekking tot auto's was geëxtraheerd. Vervolgens werd een neuraal netwerk getraind op basis van een VGG16-model, waarbij specifieke aanpassingen zijn aangebracht om het geschikt te maken voor het detecteren van bounding boxes rond auto's.

Een essentieel onderdeel van dit onderzoek was het valideren van het getrainde model. Hiervoor werd een Intersection over Union (IoU) experiment opgezet, dat diende als een maatstaf voor de nauwkeurigheid van de detecties van het model.

In deze rapportage wordt niet alleen het proces van gegevensvoorbereiding en modeltraining gedetailleerd beschreven, maar wordt ook de demonstratie van het opgezette model in een beschreven en gepresenteerd. Dit rapport belicht de bevindingen en uitdagingen gedurende dit traject en biedt inzicht in de prestaties en beperkingen van het getrainde neurale netwerk in het detecteren van auto's binnen afbeeldingen.



Figuur 1: Het verschil tussen Image Classification, Object Detection en Image Segmentation

2. Background

2.1. Evolutie van Computervisie

Computervisie heeft een opmerkelijke evolutie doorgemaakt, van traditionele methoden naar de opkomst van diepe neurale netwerken. Oorspronkelijk vertrouwde het veld op handmatig ontworpen algoritmen, maar met de opkomst van diep leren en convolutionele neurale netwerken (CNN's) is er een revolutie teweeggebracht.

De kracht van CNN's in het leren van complexe visuele patronen heeft geleid tot doorbraken in beeldclassificatie, objectdetectie en segmentatie. Dit heeft diverse domeinen beïnvloed, zoals autonome voertuigen, medische beeldvorming en veiligheidssystemen. Diepe leermodellen hebben de precisie en efficiëntie van visuele taken aanzienlijk verbeterd en hebben innovatie in computervisie gestimuleerd.

2.2. Image Classification, Object Detection en Image Segmentation

Binnen het domein van computervisie zijn verschillende taken cruciaal voor het begrijpen en analyseren van beelden: Image Classification, Object Detection en Image Segmentation.

Image Classification richt zich op het toekennen van labels aan afbeeldingen en is een fundamentele taak in computervisie. Hierbij wordt een afbeelding geclassificeerd in een bepaalde categorie of classificatie, zoals het identificeren van dieren, voertuigen of landschappen.

Object Detection daarentegen gaat verder dan alleen classificatie en omvat het lokaliseren en identificeren van (meerdere) objecten binnen een afbeelding. Dit omvat het tekenen van zogenaamde 'bounding boxes' rond objecten van interesse en het classificeren ervan.

Image Segmentation richt zich op het segmenteren van afbeeldingen in verschillende delen of segmenten, waarbij elk deel overeenkomt met een specifiek object of regio van belang. Dit resulteert in een gedetailleerde 'maskering' van elk object in de afbeelding.

Elke taak in computervisie heeft zijn eigen uitdagingen en specifieke toepassingen. Segmentatie wordt gebruikt voor medische beeldanalyse, productiedetectie en robotica. Detectie vindt zijn toepassing in videobewaking, landbouw en detailhandelsanalyse. Classificatie is waardevol voor beeldtagging, gezichtsherkenning en ziekte-diagnose. [1]

2.3. Neurale Netwerken en Transfer Learning

Neurale netwerken vormen de kern van moderne benaderingen in computervisie. Deze netwerken zijn geïnspireerd op het menselijk brein en bestaan uit lagen van neuronen die informatie verwerken en leren van data.

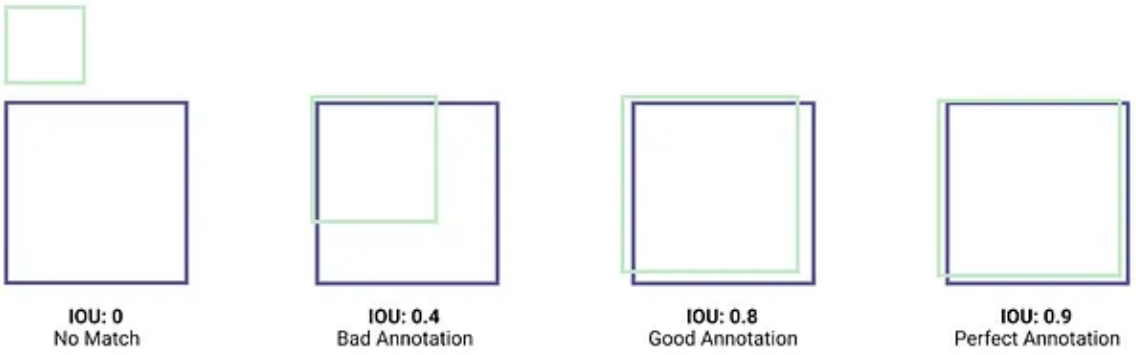
Een belangrijk concept dat bijdraagt aan het succes van neurale netwerken in nieuwe taken is "Transfer Learning". Dit is een techniek waarbij kennis die is opgedaan door een neuraal netwerk bij het oplossen van een bepaalde taak, wordt overgedragen naar een andere gerelateerde taak. In de context van computervisie kunnen vooraf getrainde modellen op grote datasets, zoals ImageNet, waardevolle kennis bevatten over verschillende visuele kenmerken. Door transfer learning kunnen deze opgedane kennis en representaties worden gebruikt als basis voor het trainen van modellen voor specifieke taken, zelfs met beperkte hoeveelheden trainingsdata. Dit heeft het mogelijk gemaakt om krachtige modellen te bouwen, zelfs in situaties waarin er geen grote datasets beschikbaar zijn. [2]

2.4. Populaire Modellen in Objectdetectie, inclusief VGG16

Binnen objectdetectie vormen modellen zoals YOLO (You Only Look Once) [3], SSD (Single Shot MultiBox Detector) [4], Faster R-CNN (Region-based Convolutional Neural Network) [5] en VGG16 (Visual Graphics Group 16) [6] enkele van de meest gerespecteerde en gebruikte modellen. Hoewel VGG16 in eerste instantie is ontwikkeld voor beeldclassificatie, wordt het ook vaak gebruikt als basis voor detectiemodellen vanwege zijn architecturale eenvoud en effectiviteit in het leren van representaties van beelden. De andere modellen variëren in hun aanpak, waarbij sommige zich richten op real-time detectie, terwijl andere prioriteit geven aan nauwkeurigheid. De impact van deze modellen strekt zich uit over verschillende toepassingsgebieden, waaronder autonome voertuigen, veiligheidssystemen en objectherkenningstoepassingen.

2.5. Technieken voor Objectdetectiemodelvalidatie

Binnen objectdetectie in computervisie worden diverse methoden gebruikt om de nauwkeurigheid van modellen te valideren. Een van de technieken is de Intersection over Union (IoU) te zien in figuur 2, die de mate van overlapping tussen voorspelde en daadwerkelijke objectgrenzen meet. Andere veelgebruikte methoden zijn precision, recall en F1-score, die verschillende aspecten van de modelprestaties belichten, zoals nauwkeurigheid en volledigheid van de detecties. Het gebruik van evaluatietechnieken is cruciaal om de effectiviteit en betrouwbaarheid van objectdetectiemodellen te beoordelen en vormt een essentiële stap bij het verbeteren van deze modellen voor praktische toepassingen. [7]



Figuur 2: Intersection over Union (IoU)

3. Methodebeschrijving

In eerste instantie heb ik mijn dataset geanalyseerd, omdat dit van invloed zal zijn op de toplagen. De dataset is een single-class dataset geannoteerd met bounding box labels. Daarna heb ik verschillende modellen overwogen en besloten om VGG16 te gebruiken vanwege zijn eenvoud in architectuur en effectiviteit. Echter, vanwege de oorspronkelijke geschiktheid van het model voor objectclassificatie, moest ik de laatste lagen van het model aanpassen. Daarom ben ik aan de slag gegaan met een tutorial van Marius over Transfer Learning, waarin stapsgewijs werd uitgelegd hoe dit principe toe te passen is. Nadat ik de tutorial had gevolgd had ik een goed beeld over transfer learning en heb ik dus een custom model kunnen maken voor mijn use-case gebaseerd op VGG16 van Keras.

3.1. Dataset inladen en verwerken

Ik heb een functie geschreven te zien in figuur 3, om de geselecteerde dataset gemakkelijk te laden voor gebruik met het VGG16-model. Deze functie laadt elke foto in het verwachte formaat van het VGG16-model (224x224x3), past normalisatie toe en haalt vervolgens de bijbehorende bounding box-informatie uit de bijbehorende csv-bestanden. Daarna schaal ik de bounding box-informatie op dezelfde manier als de afbeeldingen en voeg ik het toe aan de juiste numpy-arrays.

```
def load_dataset(dataset):
    processed_images = []
    extracted_bboxes = []

    for item in os.listdir(dataset):
        if item.endswith('.jpg'):
            img_path = os.path.join(dataset, item)

            # Load the image and resize to 224x224
            img = img_to_array(load_img(img_path, target_size=(224, 224)))
            normalized_img = img.astype('float32') / 255.0  # Normalize image to retain original color
            processed_images.append(normalized_img)

            # Extract bounding box information from corresponding XML file
            xml_filename = os.path.splitext(item)[0] + '.xml'
            tree = ET.parse(os.path.join(dataset, xml_filename))
            root = tree.getroot()
            bndbox = root.find('.//bndbox')
            xmin, ymin, xmax, ymax = map(int, [bndbox.find(x).text for x in ['xmin', 'ymin', 'xmax', 'ymax']])
            img_shape = Image.open(img_path).size
            scale_x, scale_y = 224 / img_shape[0], 224 / img_shape[1]
            scaled_xmin, scaled_ymin, scaled_xmax, scaled_ymax = map(int, [xmin * scale_x, ymin * scale_y, xmax * scale_x, ymax * scale_y])
            extracted_bboxes.append([scaled_xmin, scaled_ymin, scaled_xmax, scaled_ymax])

    return np.array(processed_images), np.array(extracted_bboxes)
```

Figuur 3: Functie die de gegeven dataset inlaad en verwerkt.

3.2. Het Model

Het aangepaste model te zien in figuur 4, is gebouwd op basis van het Keras VGG16-model, een convolutioneel neuraal netwerk dat vaak wordt gebruikt in taken voor beeldherkenning. Dit aangepaste model heeft als doel één specifieke klasse binnen afbeeldingen te detecteren door het VGG16-model opnieuw te trainen met een specifieke dataset. Het proces omvat het vervangen van de laatste verbonden laag door een nieuwe uitvoerlaag die is ontworpen om de bounding box-coördinaten van het beoogde object in de afbeelding te voorspellen.

De code in figuur 4, toont de stappen die zijn genomen om het VGG16-model af te stemmen. De eerste actie omvat het laden van het VGG16-model zonder de topclassificatielagen. Vervolgens worden aanpassingen aangebracht door een Flatten-laag toe te voegen om de uitvoer om te zetten naar een eendimensionale tensor. Daarna wordt een Dense-laag met 256 units en een ReLU-activatiefunctie toegevoegd. De laatste aangepaste laag bestaat uit 4 units en maakt gebruik van een lineaire activatiefunctie om voorspellingen te genereren voor de bounding box-coördinaten. Vervolgens wordt het model gecompileerd met de Adam-optimizer.

```python
# Load VGG16 model without the top classification layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=[224, 224, 3])

# Flatten the output and add custom output layers
x = Flatten()(base_model.output)
x = Dense(256, activation='relu')(x)

# Output layer for object detection
bbox_output = Dense(4, activation='linear')(x) # Bounding box coordinate predictions

# Create a new model with custom input and output layer
Custom_VGG16 = Model(inputs=base_model.input, outputs=bbox_output)

# Freeze the weights of the pre-trained layers
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
Custom_VGG16.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_squared_error'])
```

Figuur 4: Custom VGG16 d.m.v. Transfer Learning.

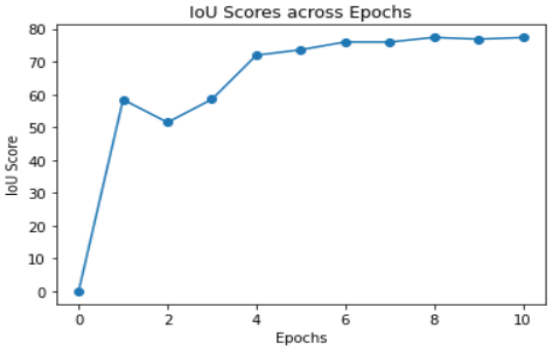
4. Experiment(en)beschrijving

Om de prestaties van het custom VGG16-model te evalueren, werd een experiment uitgevoerd waarbij testafbeeldingen en hun bounding box labels werden vergeleken met de uitvoer van het model. Deze evaluatie berekende Intersection over Union (IoU) scores voor elke afbeelding en vervolgens de IoU score over de gehele test set.

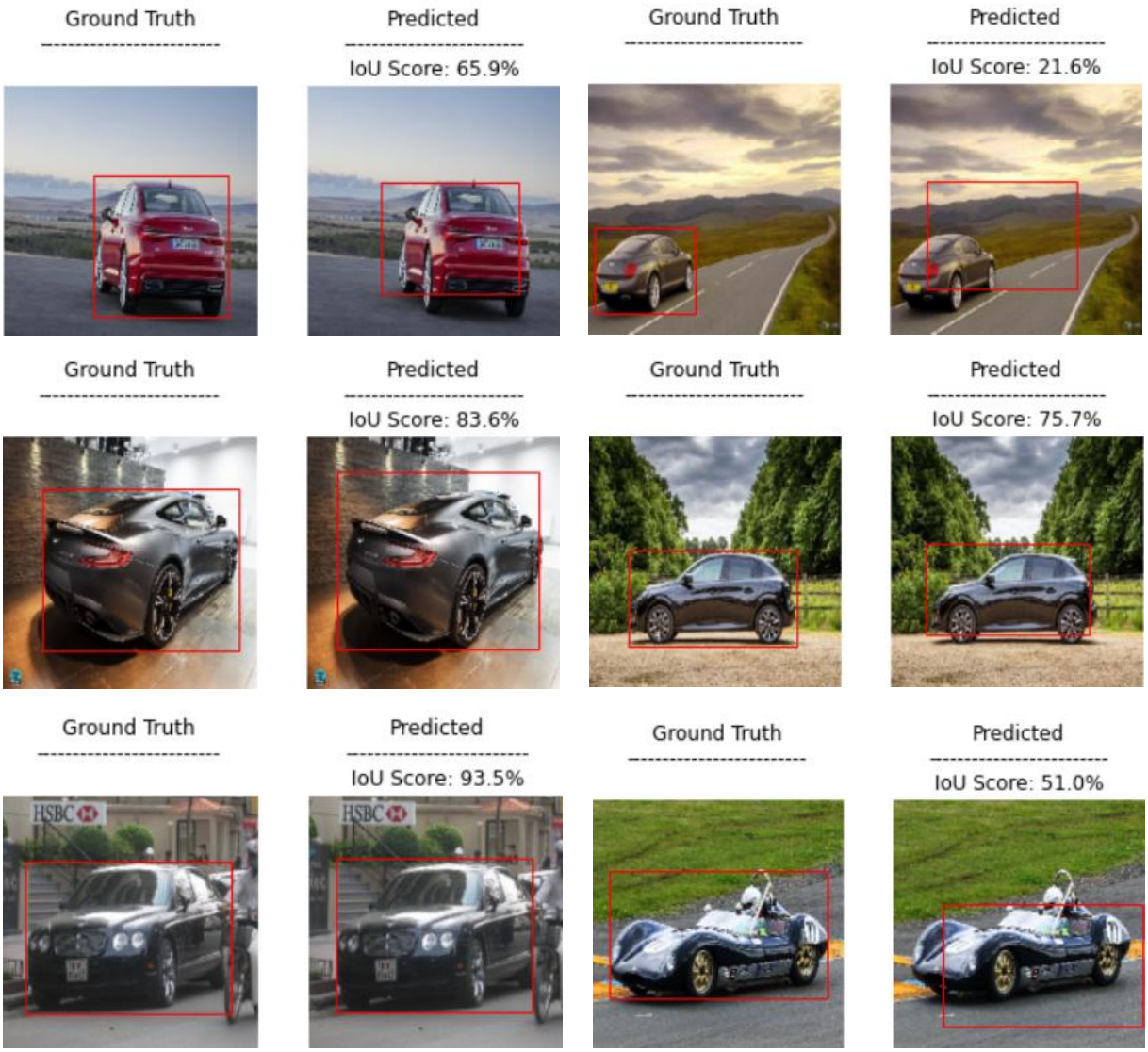
Dit validatieproces vergemakkelijkte niet alleen visuele vergelijkingen tussen voorspellingen van het model en de werkelijke labels, maar kwantificeerde ook nauwkeurigheid met behulp van IoU-scores, waardoor uitgebreide inzichten werden verkregen in de dataset brede prestaties van het model.

Het model blinkt uit in het nauwkeurig detecteren van auto's uit afbeeldingen, vooral in scenario's met slechts één object, wat overeenkomt met het ontwerp als een model voor single box detectie. Uitdagingen ontstaan wanneer meerdere objecten aanwezig zijn in een afbeelding of wanneer de lucht donkergrijs is, wat leidt tot een algemene daling in de IoU-score. Momenteel behaalt het model na tien epochs een totale IoU-score van 77,4%.

Wat betreft de selectie van epochs, een analyse van de trends in verlies en IoU-nauwkeurigheid tijdens de training onthulde dat de trend in verlies minimale invloed had op het besluitvormingsproces. Bijgevolg verschoof de nadruk alleen naar de IoU-score om het optimale aantal epochs te bepalen. De beslissing werd genomen om het model gedurende tien epochs te trainen. Grafische voorstellingen geven duidelijk aan dat de IoU-nauwkeurigheid plateauvormig wordt na ongeveer acht epochs, wat wijst op afnemende verbeteringen in nauwkeurigheid voorbij dit punt.



Figuur 5: Nauwkeurigheid over epochs



Figuur 6: Visualisatie van de IoU tests.

5. Conclusie/Discussie

De resultaten laten zien dat het model effectief is in het detecteren van voertuigen in verschillende omgevingen en geschikt is voor objectdetectie. Over een testset van 57 foto's behaalde het model een gemiddelde IoU-score van 77,4%, wat aangeeft dat het over het algemeen nauwkeurige detecties uitvoert. Echter, het model heeft ook zijn beperkingen. Het presteert bijvoorbeeld minder goed bij grijze luchten en kan slechts één object per afbeelding detecteren vanwege de single-box detection output in plaats van multi-box detection. Bovendien traint het model momenteel op de processor, wat resulteert in beperkte snelheid in vergelijking met een grafische kaart. Voor toekomstig onderzoek wordt aanbevolen om het algoritme te testen op grotere datasets en de prestaties te optimaliseren voor real-time toepassingen.

6. Literatuurlijst

[1] Picsellia, „Segmentation vs Detection vs Classification in Computer Vision: A Comparative Analysis," 29 Mei 2023. [Online]. Available: https://www.picsellia.com/post/segmentation-vs-detection-vs-classification-in-computer-vision-a-comparative-analysis. [Geopend 9 December 2023].

[2] P. Sharma, „Understanding Transfer Learning for Deep Learning," 7 December 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/. [Geopend 9 December 2023].

[3] R. J. F. en A. , „YOLO: Real-Time Object Detection," 2018. [Online]. Available: https://pjreddie.com/darknet/yolo/. [Geopend 10 December 2023].

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu en A. C. Berg, „1512.02325.pdf," 29 December 2016. [Online]. Available: https://arxiv.org/pdf/1512.02325.pdf. [Geopend 11 December 2023].

[5] S. Ren, K. He, R. Girshick en J. Sun, „1506.01497.pdf," 6 Januari 2016. [Online]. Available: https://arxiv.org/pdf/1506.01497.pdf. [Geopend 11 December 2023].

[6] K. Simonyan en A. Zisserman, „1409.1556.pdf," 2015. [Online]. Available: https://arxiv.org/pdf/1409.1556.pdf. [Geopend 11 December 2023].

[7] I. Tan, „Measuring Labelling Quality with IOU and F1 Score," 24 Maart 2020. [Online]. Available: https://medium.com/supahands-techblog/measuring-labelling-quality-with-iou-and-f1-score-1717e29e492f. [Geopend 11 December 2023].

7. Bijlage

In de bijlage geef ik beknopte uitleg per beoordelingscriterium, waarbij ik mijn acties en benaderingen illustreer met links of verwijzingen naar de specifieke secties van mijn onderzoeksrapport.

7.1. Vision-Algoritmes implementeren

Het geïmplementeerde vision-algoritme is te vinden in mijn NoteBook op GitHub en wordt daarnaast ook toegelicht in hoofdstuk 3: Methodebeschrijving van het Eindrapport.

7.2. Planning

Ik heb bewust gekozen om het wekelijks bijhouden van activiteiten in een document niet toe te passen, nadat ik dit besprak met mijn docent. In plaats daarvan heb ik regelmatig en direct met mijn docent over mijn voortgang gesproken. Onze frequente gesprekken boden een effectievere manier om directe feedback te ontvangen en mijn aanpak flexibel aan te passen. Hoewel ik geen specifiek document bijhield, bood deze interactie een waardevolle en directe manier om mijn voortgang te monitoren en verbeteren.

7.3. Versiebeheer (GIT)

Tijdens mijn project heb ik actief GIT gebruikt om mijn werk bij te houden en belangrijke wijzigingen te registreren. Ik heb regelmatig 'commits' gedaan, die te vinden zijn op GitHub.

7.4. Experiment opzet

Voor dit project heb ik een experiment opgezet in de vorm van een Custom-VGG16 model dit model en de effectiviteits-testen hiervan zijn te vinden in mijn NoteBook op GitHub. De grenzen van mijn model worden tevens uitgelegd in mijn NoteBook op GitHub en in Hoofdstukken 3 en 4 van het Eindrapport.

7.5. Eindrapport

Voor dit project heb ik dit rapport opgezet, waarin ik het gekozen en uitgevoerde onderzoeken, experiment en resultaten beschrijf en tot slot een conclusie trek.

7.6. Experiment resultaten

De resultaten van mijn experimenten zijn ook te vinden in mijn NoteBook op GitHub en daarnaast bespreek ik ze ook in Hoofdstuk 4 en 5 van het Eindrapport.