

CAPTCHA Resolver

Onderzoeksverslag

Versie: 1.0

Door: Mike van Egmond (646991)
Auke Onvlee (604640)
Steven Velderman (636657)
Alex Cheng (634967)

Groep: 4

Klas: ITA-NotS-A-f
Docent: Theo Theunissen & Helen Visser
Datum: 14-02-2021
Course: NOTS Project

1 Inleiding	4
2 Probleemstelling	5
2.1 Wat is de huidige situatie?	5
2.2 Wat is de gewenste situatie?	5
2.3 Wat is het verschil tussen de huidige en gewenste situatie?	5
3 Doelstelling	5
4 Vraagstelling	6
4.1 Hoofdvraag	6
4.2 Deelvragen	6
4.2.1 Wat is Deep Learning?	6
4.2.1.1 Motivatie	6
4.2.1.2 Relevantie	6
4.2.1.3 Onderzoeksmethode	6
4.2.2 Welke Deep Learning technieken zijn er nodig om dit probleem op te lossen?	6
4.2.2.1 Motivatie	6
4.2.2.2 Relevantie	6
4.2.2.3 Onderzoeksmethode	7
4.2.3 Welke data hebben we nodig om het model te kunnen trainen?	7
4.2.3.1 Motivatie	7
4.2.3.2 Relevantie	7
4.2.3.3 Onderzoeksmethode	7
4.2.4 Hoe zeker moet het model zijn over zijn keuze om aan te geven welke letter correct is?	7
4.2.4.1 Motivatie	7
4.2.4.2 Relevantie	7
4.2.4.3 Onderzoeksmethode	8
4.2.5 Hoe kunnen we de accuraatheid van het model verbeteren?	8
4.2.5.1 Motivatie	8
4.2.5.2 Relevantie	8
4.2.5.3 Onderzoeksmethode	8
4.2.6 Wat zijn de ethische gevolgen van het automatiseren van CAPTCHA's?	8
4.2.6.1 Motivatie	8
4.2.6.2 Relevantie	8
4.2.6.3 Onderzoeksmethode	9
5 Resultaten	10
5.1 Wat is Deep Learning?	10
5.1.1 Artificial Intelligence en Machine Learning (Trendskout, 2021)	10
5.1.2 Deep Learning	10
5.1.3 Layers	11
5.2 Welke Deep Learning technieken zijn er nodig om dit probleem op te lossen?	12
5.2.1 Neural Network	12

5.2.2.1 CNN	12
5.2.2 Tools om te trainen	13
5.2.2.1 CPU, GPU of TPU:	14
5.2.2.1.1 CPU	14
5.2.2.1.2 GPU	14
5.2.2.1.3 TPU	14
5.2.2.1.4 Uiteindelijke keuze	14
5.2.2.2 Google Colab / Kaggle	14
5.2.3 Transfer learning	15
5.2.3.1 COCO	16
5.2.3.2 EfficientDet	16
5.2.3.3 YOLOv5	17
5.2.3.3.1 Pre-trained checkpoints & Models	18
5.2.3.3.2 Hyperparameters	18
5.2.3.4 Faster R-CNN	19
5.2.3.5 EfficientDet (TensorFlow)	20
5.2.3.6 Mask R-CNN	21
5.3 Welke data hebben we nodig om het model te kunnen trainen?	22
5.3.1 Data Augmentation	22
5.3.1.1 Verschillende soorten data augmentation	23
5.3.1.1.1 Noise	23
5.3.1.1.2 Cutout	24
5.3.1.1.3 Shear	24
5.3.1.1.4 Rotate	25
5.3.1.2 Toepassen van data augmentation	25
5.4 Hoe zeker moet het model zijn over zijn keuze om aan te geven welke letter correct is?	28
5.4.1 Confidence score	28
5.4.2 Precision en Recall	29
5.5 Hoe kunnen we de accuraatheid van het model verbeteren?	30
5.5.1 Balans in dataset	30
5.5.2 Precision	31
5.5.3 Recall	33
5.6 Wat zijn de ethische gevolgen van het automatiseren van CAPTCHA's?	37
5.6.1 Ethische theorieën	37
5.6.2 Deugdethiek	37
5.6.3 Deontologie	38
5.6.4 Utilitarisme	38
6 Discussie	40
6.1 Niet behaald	40
6.1.1 Gespiegelde letters	40
6.1.2 Alternatieve modellen	40
6.1.3 Datasets	40
6.2 Wel behaald	40

6.2.1 Werkend model	40
6.2.2 Bevindingen	41
6.2.3 Onderzoeksverslag	41
6.3 Meegenomen kennis	41
6.3.1 Voortzetten van trainen	41
6.3.2 Data Augmentation	41
6.3.3 Transfer Learning	41
6.3.4 Diverse Tools (Wandb/Roboflow/Google Colab)	42
6.3.4.1 WandB	42
6.3.4.2 Roboflow	42
6.3.4.3 Google Colab	42
6.4 Onverwachte veranderingen	42
6.4.1 Labelen van data	42
6.4.2 Van Tensorflow naar PyTorch	42
6.4.3 Bevindingen	43
7 Conclusie	44
7.1 Proof of Concept	44
7.1.1 Eigengemaakte woord	44
7.1.2 Eigengemaakte woord met strepen erdoorheen.	45
7.1.3 Meerdere CAPTCHA's in 1 plaatje.	45
Alle Google Colabs	46
Models	46
Trainen	46
Detecteren	46
Literatuurlijst	47

1 Inleiding

Hedendaags merken we dat wij op bijna iedere website wel te maken hebben met zogenaamde CAPTCHA's. Hierbij hebben wij soms wel moeite om deze fatsoenlijk op te lossen. Uit onderzoek (Scott, z.d.) is ook gebleken dat wij niet de enige zijn, dus daarom lijkt het ons leuk om dit probleem te gaan uitzoeken en proberen op te lossen. Vandaar dat wij dit onderzoeksverslag hebben opgesteld.

In dit onderzoeksverslag documenteren we ons onderzoek naar het automatisch oplossen van CAPTCHA's met behulp van Deep Learning. We hebben hierbij rekening gehouden met de voor- en nadelen van deze implementatie zoals bijvoorbeeld de ethische kant. Denk hierbij aan hackers of phishers die hedendaags steeds meer en meer gevreesd worden. Ondanks dat is het automatisch oplossen van CAPTCHA's erg relevant, omdat we merken dat CAPTCHA's steeds vaker worden gebruikt op websites. Daarnaast worden ze tegenwoordig ook lastiger.

2 Probleemstelling

2.1 Wat is de huidige situatie?

In de huidige situatie lossen mensen CAPTCHA's op door een aantal letters over te typen. Het doel van CAPTCHA's is om robots tegen te houden en zo de veiligheid van een website of applicatie te vergroten.

Meer dan 33% (Boag, 2017) van de gebruikers faalt bij de eerste poging van een CAPTCHA. Dit is dus frustrerend voor de gebruiker omdat het erg tijdrovend is om opnieuw deze CAPTCHA's op te lossen. Daarnaast is er ook gebleken dat het slecht voor een bedrijf kan zijn om CAPTCHA's te benutten, het jaagt bijna 10% van de gebruikers weg. (Boag, 2017)

2.2 Wat is de gewenste situatie?

In de gewenste situatie hoeven de letters van een CAPTCHA niet meer handmatig overgetypt te worden. De letters worden door de computer herkend door middel van Deep Learning en automatisch ingevuld voor de gebruiker.

2.3 Wat is het verschil tussen de huidige en gewenste situatie?

Er is geen kant en klaar programma dat de aan ons aangeleverde CAPTCHA's kan herkennen. Deep Learning is een techniek die toegepast kan worden om deze CAPTCHA's te gaan herkennen. Het is echter onduidelijk hoe de CAPTCHA's opgelost kunnen worden met behulp van deze techniek. Daarbij is het verschil tussen de huidige situatie en de gewenste situatie dat hedendaags een CAPTCHA nog handmatig opgelost moet worden, terwijl wij dit nu willen gaan automatiseren. Verder zal de veiligheid die CAPTCHA's met zich meebrengen, teniet gedaan worden als deze automatisch opgelost worden. Het is de vraag in hoeverre dat ethisch verantwoord is.

3 Doelstelling

Het doel van dit onderzoek is om inzicht te krijgen over de werking van Deep Learning en hoe dit toegepast kan worden om CAPTCHA's automatisch op te lossen.
Dit doel vormt de basis van onze hoofdvraag. Gedurende 16 weken gaan we door middel van het beantwoorden van zes deelvragen, de hoofdvraag beantwoorden. Deze worden allemaal onderzocht door middel van diverse experimenten.

4 Vraagstelling

4.1 Hoofdvraag

We doen dit onderzoek om antwoord te geven op de volgende vraag:

Hoe kunnen we CAPTCHA's automatisch oplossen met behulp van Deep Learning?

4.2 Deelvragen

Om deze vraag te kunnen beantwoorden hebben we hem onderverdeeld in meerdere deelvragen. Door deze deelvragen te beantwoorden kunnen we antwoord geven op de hoofdvraag.

4.2.1 Wat is Deep Learning?

4.2.1.1 Motivatie

Voor dat we aan de slag kunnen gaan met het zoeken naar een oplossing van het probleem moeten we ons eerst inlezen in Deep Learning. Wat is het precies en waarom zouden we Deep Learning gebruiken?

4.2.1.2 Relevantie

Als je niet bekend bent met de termen zal het ook lastig zijn om online bronnen te kunnen gebruiken. We zullen namelijk moeten weten waar we mee werken om het toe te kunnen passen en aan te kunnen passen.

4.2.1.3 Onderzoeks methode

Om dit uit te zoeken gaan we gebruik maken van Library. Er is namelijk online veel informatie te vinden over Deep Learning.

4.2.2 Welke Deep Learning technieken zijn er nodig om dit probleem op te lossen?

4.2.2.1 Motivatie

Er bestaan heel veel verschillende soorten neurale netwerken die gebruikt kunnen worden voor het trainen van data. Ieders hebben zijn eigen voordelen dus is het belangrijk om te onderzoeken welke het meest geschikt is voor ons.

4.2.2.2 Relevantie

Wij willen ons model het meest accuraat mogelijk maken door de meest efficiënte Deep Learning techniek te gebruiken. Hierbij is vooral de accuraatheid belangrijk voor onze usecase. Echter wanneer wij de accuraatheid willen verbeteren dan gaat het ten koste van de snelheid. Deze snelheid is niet heel relevant voor ons want we trainen en/of detecteren geen video's, maar stilstaande afbeeldingen.

4.2.2.3 Onderzoeks methode

We gebruiken voor deze methode Library. Er is al veel onderzoek gedaan naar Deep Learning dus kunnen we die kennis gebruiken om sneller aan de slag te kunnen gaan en gefocust te kunnen gaan werken.

4.2.3 Welke data hebben we nodig om het model te kunnen trainen?

4.2.3.1 Motivatie

Voor het trainen van ons model gaan we data gebruiken. We hebben in dit geval een uitgebreide dataset aangeleverd gekregen. Deze data zullen wij als groep verdelen om uit te zoeken wat precies bruikbaar en inzetbaar is. Daarbij kijken wij naar het toevoegen van extra data, weghalen van data en het aanpassen van de dataset aan de hand van tooling. Om erachter te komen welke dataset bruikbaar is om het model te trainen, zal er ook geëxperimenteerd worden met verschillende datasets.

Aan de hand van gegenereerde grafieken kan er afgelezen worden welke data het meest effectief werkt. Hiervoor gebruiken we Loss functions, precision en recall. Daarnaast kunnen we ook zelf oordelen op basis van de output van het model. Deze dataset zal vervolgens gelabeld worden, zodat het model kan herkennen welke individuele letters er zitten in een afbeelding. Ten slotte gaan we onderzoeken welke structuur de dataset zal gaan krijgen.

4.2.3.2 Relevantie

Om het model goed te kunnen trainen en de letters individueel te kunnen herkennen hebben we de juiste data nodig. De juiste data is in dit geval een dataset dat effectief werkt bij het trainen van ons model en ervoor zorgt dat letters individueel herkend kunnen worden.

4.2.3.3 Onderzoeks methode

De onderzoeks methode die wij gaan gebruiken is: "Library". Met deze onderzoeks methode kunnen we onderzoeken welke data benodigd is en wat het juiste output formaat zal zijn voor de data.

4.2.4 Hoe zeker moet het model zijn over zijn keuze om aan te geven welke letter correct is?

4.2.4.1 Motivatie

We zijn geïnteresseerd of het resultaat verschilt wanneer de AI zekerder of onzekerder is. Hiermee wordt bedoeld voor hoeveel procent het model aangeeft welke letter er staat en of deze letter ook klopt. Op het moment dat een letter bijvoorbeeld voor honderd procent wordt herkend, dan is het model zeker welke letter er staat. Wij willen uitzoeken wanneer dit resultaat verandert gebaseerd op een lagere of hogere accuraatheid van het model.

4.2.4.2 Relevantie

Wij willen uitzoeken op welk moment het model zeker is welke individuele letter staat in de CAPTCHA afbeelding. Dit wordt door het model uitgedrukt in een percentage van zekerheid. Wij willen onderzoeken bij welke letters in de dataset het model met zekerheid kan zeggen

dat het een bepaalde letter is, en wanneer dit ook daadwerkelijk correct is. Het is interessant om te onderzoeken welke letters met een hogere of lagere zekerheid herkend worden, en wat de redenen hiervoor zijn.

4.2.4.3 Onderzoeksmethode

Voor deze deelvraag maken wij gebruik van twee onderzoeksmethoden. Om te onderzoeken hoe andere AI projecten aantonen hoe accuraat een model is, maken we gebruik van: "Library". Daarnaast maken wij gebruik van de onderzoeksmethode: "Lab", om te onderzoeken op welke manier ons model accurater wordt.

4.2.5 Hoe kunnen we de accuraatheid van het model verbeteren?

4.2.5.1 Motivatie

We willen gaan kijken naar welke aspecten van belang zijn voor het verhogen van de accuraatheid. Een richting zou kunnen zijn meer data, of beter gelabelde data. Een trade-off waar we naar zullen kijken is accuraatheid in vergelijking met snelheid. Als de accuraatheid hoger is, zorgt dat er voor dat het trainen langer duurt? En hoe groot is het verschil tussen het eenmalige trainen in tegenstelling tot het meerdere keren gebruiken?

4.2.5.2 Relevantie

Een hogere accuraatheid zal beter zijn voor de eindgebruiker omdat de CAPTCHA's dan vaker kunnen worden opgelost.

4.2.5.3 Onderzoeksmethode

Hierbij willen we de onderzoeksmethode "Lab" toepassen, want we willen gebruik maken van testen om te controleren welke verschillende variabelen invloed heeft op bijvoorbeeld onze data of het model.

4.2.6 Wat zijn de ethische gevolgen van het automatiseren van CAPTCHA's?

4.2.6.1 Motivatie

Ethisch is nadenken over iets moreel goed is om te doen (Ensie, 2022), en daarom is het erg belangrijk om hierbij stil te staan. In ons geval zijn we bewust dat er ethische bezwaren zijn bij het realiseren van het probleem dat we willen oplossen, denk hierbij aan hackers die zonder consequenties constant een wachtwoorden kunnen gaan kraken zonder last te hebben van een CAPTCHA ervoor. Hierbij willen wij ervoor zorgen dat we onderzoeken wat de ethische gevolgen zijn en dat we er bij stil hebben gestaan.

4.2.6.2 Relevantie

Hedendaags wordt hacken steeds populairder en lastiger te voorkomen (Hendrickson, 2022). Daarom kan het automatisch oplossen van CAPTCHA's bij de verkeerde doelgroep ook misbruikt worden.

4.2.6.3 Onderzoeks methode

Wij willen de onderzoeks methode “Field” toepassen, omdat we door middel van survey’s willen peilen wat volgens andere doelgroepen en/of mensen de meest belangrijke ethische gevolgen zijn.

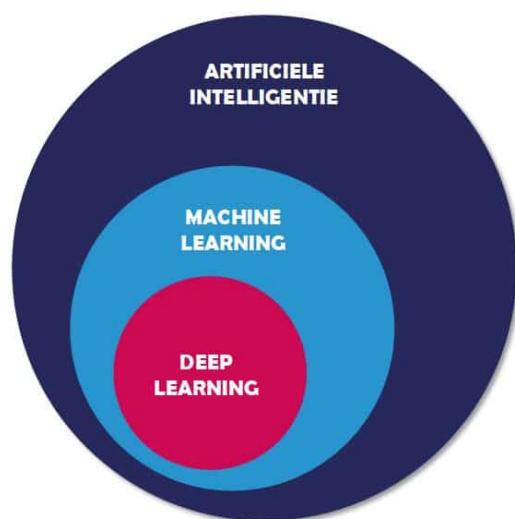
5 Resultaten

5.1 Wat is Deep Learning?

Voordat we kunnen uitleggen wat Deep Learning is, is het handig om te weten wat Machine Learning en Artificial Intelligence is.

5.1.1 Artificial Intelligence en Machine Learning

Artificial Intelligence (IBM Cloud Education, 2021), oftewel AI, is de wetenschap achter het maken van een intelligent machine, voornamelijk computer programma's. Het is ontwerpen om de menselijke intelligentie te begrijpen en na te werken (Oracle, z.d.). Hoe deze menselijke intelligentie wordt ontwikkelt gaan we benoemen in de onderstaande hoofdstukken.

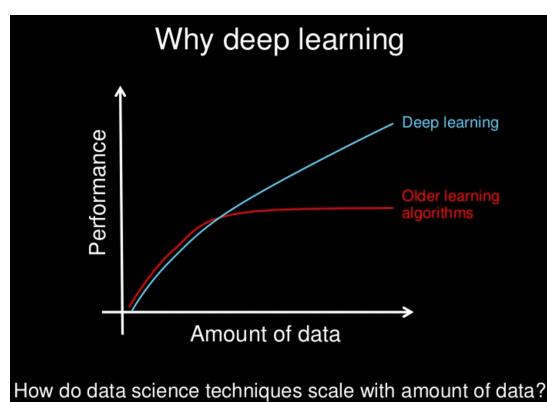


Binnen AI kennen we ook Machine Learning, ML in het kort. ML is de studie van een computer algoritme, deze zal zichzelf ontwikkelen na ervaringen en door middel van data.

Figuur 1 (Trendskout, 2021)

5.1.2 Deep Learning

Deep Learning is een subcategorie van Machine Learning. Deep Learning is een neuraal netwerk met minimaal 3 lagen (Education, 2022). Er wordt met Deep Learning geprobeerd om het gedrag van de mens na te bootsen/simuleren. De benodigde hoeveelheid gelabelde data is bij Deep Learning veel hoger.



Figuur 2 (Ng, 2019)

Vergelijkbaar met Deep Learning verschilt het nogal van Machine Learning (Grossfeld, 2020). Het grootste verschil hiertussen is de manier van hoe de AI het algoritme zich uitbreidt. Bij ML kijken we naar de data, waarbij er specifiek gericht wordt naar de gemaakte keuze gebaseerd wat er geleerd is (Oracle, z.d.).

Eigenlijk zou je DL kunnen zien als een onderdeel van ML. Beide vallen ze onder de globale term Artificial Intelligence, echter verschilt het resultaat.

Ten slotte zijn er binnen Machine Learning twee manieren van trainen. De meest gebruikte methode is Supervised Machine Learning, hierbij gebruik je een gelabelde dataset om het model te trainen. Unsupervised Machine Learning is het tegenovergestelde. Hierbij laat je het model zelf iets proberen te interpreteren.

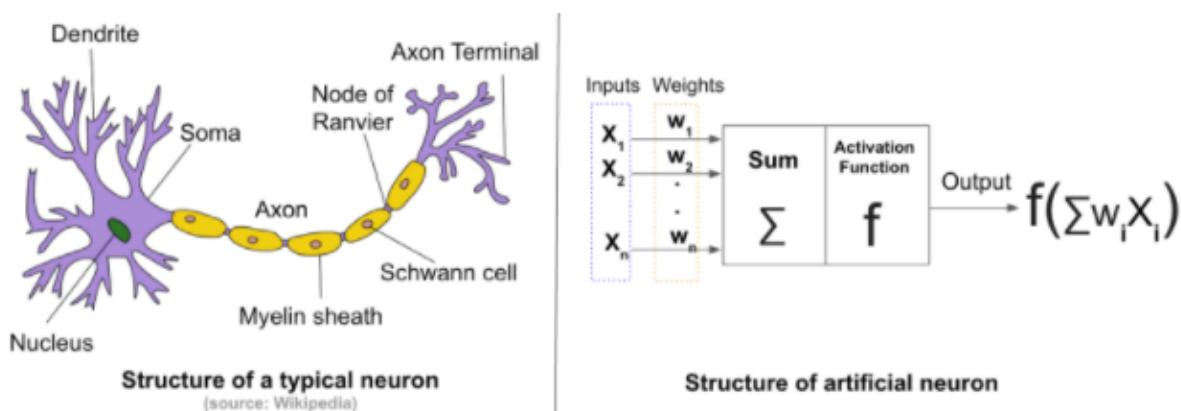
5.1.3 Layers

Het wordt ‘Deep’ verwijst naar de meerdere lagen in het netwerk. Een laag is een stap binnen het proces wat het model neemt. Het netwerk bestaat uit meerdere typen lagen. Zo is er een input-laag, waar de input inkomt. Dan zijn er de ‘verborgen’ lagen, deze zorgen voor de intelligentie van het netwerk. En ten slotte is er de output-laag (Rooijakkers, 2022).

5.2 Welke Deep Learning technieken zijn er nodig om dit probleem op te lossen?

5.2.1 Neural Network

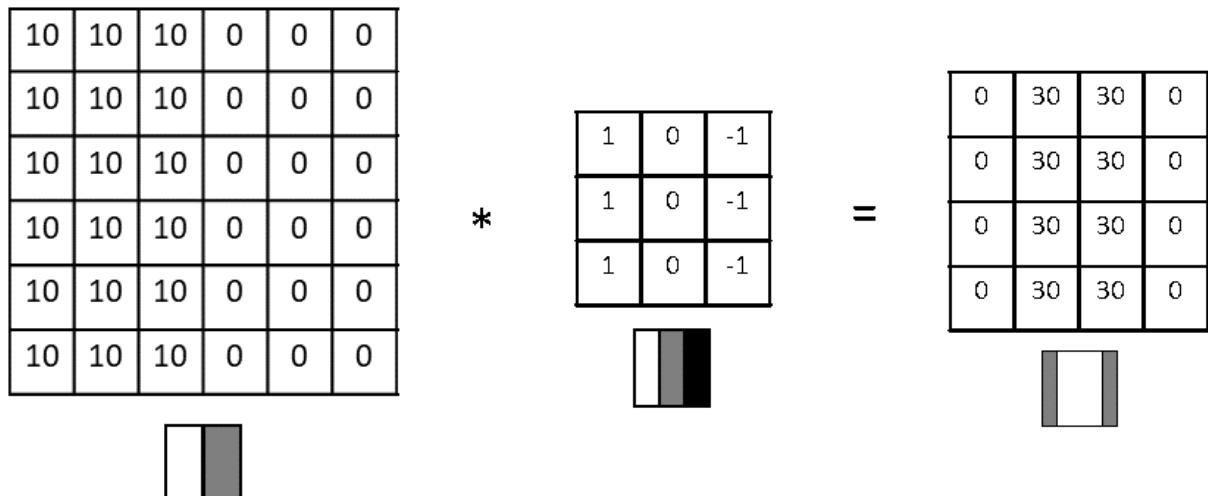
Een neuraal netwerk is een serie van algoritmes, dat probeert om de werking van het brein te simuleren/na te bootsen. Dit kunstmatige 'brein' ontwikkelt zijn model door te communiceren tussen Artificial Neurons. Zo'n neuron ontvangt een of meerdere inputs om daarna een output terug te geven. Een Artificial Neural Network, oftewel ANN, is hierdoor dus ook een collectie van vele neuronen bij elkaar. Binnen een neuron wordt een bepaalde functie uitgevoerd. Dit is de activation function.



Figuur 3 (Man vs machine: comparing artificial and biological neural networks, 2017)
De Myelin Sheath en Schwann cell zitten niet in een artificial neuron.

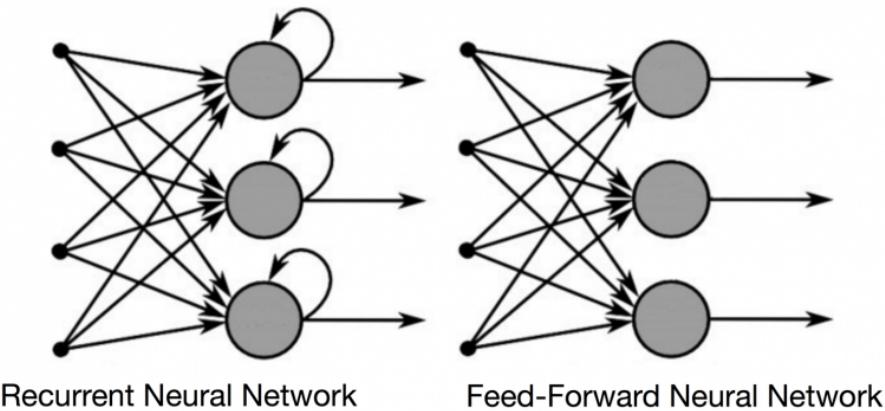
5.2.2 CNN

Een CNN is een 'Convolutional Neural Network'. Dit is een neuraal netwerk met een convolutionele laag. Een convolutionele laag kan kenmerken van een afbeelding herkennen. Een voorbeeld van zo'n kenmerk zijn de randen van objecten in een afbeelding. Deze worden herkend door middel van 'Edge Detection'. Bij edge Detection wordt een afbeelding vereenvoudigd tot een matrix van cijfers. Deze cijfers representeren de helderheid van de pixels. Er wordt gebruik gemaakt van Convolution om veranderingen in de helderheid te detecteren. Hierbij gaat een kernel (of filter) stap voor stap over de matrix heen, waar weer een nieuwe matrix uitkomt (Nkomo, 2022). In de afbeelding hieronder is een voorbeeld van verticale randen detectie te zien. Hierbij is de originele matrix de meest linker deel van het plaatje, de kernel het middelste deel en de uitkomst matrix het rechter deel. Hoe kleiner de kernel is, des te groter wordt de uitkomst matrix en de nauwkeurigheid van de randen detectie. Een kleinere stapgrootte (stride) zorgt ook voor een grotere uitkomstmatrix en dus een hogere nauwkeurigheid (DeepAI, 2020).



Figuur 4 (#003 CNN More On Edge Detection, 2018)

Naast CNN's zijn er ook RNN's, dit staat voor Recurrent Neural Networks (Telus, 2022). In tegenstelling tot CNN's, zijn RNN's goed in het begrijpen van tekst en spraak. Bijvoorbeeld bij tekst is het belangrijk om de context van een zin te hebben. Een bepaald woord kan namelijk een andere betekenis hebben afhankelijk van de context. Voor onze use case is het gebruiken van een RNN daarom niet relevant.



Figuur 5 (A Guide to RNN: Understanding Recurrent Neural Networks and LSTM Networks, 2021)

5.2.2 Tools om te trainen

Bij het trainen van een AI model kan gebruik gemaakt worden van verschillende tooling. Er is daarbij een onderscheid tussen de hardware en software dat benodigd is. In dit hoofdstuk wordt uitgelegd welke hardware nodig is om een model te kunnen trainen en wat het effect daarvan is. Daarnaast wordt er uitgelegd welke software gebruikt is om een AI model te trainen, en welke overwegingen zijn gemaakt bij het kiezen van een type 'notebook'.

5.2.2.1 CPU, GPU of TPU:

Bij het trainen van een model kan er gebruikt gemaakt worden van een CPU, GPU en TPU. De werking van deze hardware is verschillend en niet elke hardware heeft hetzelfde effect bij het trainen van een AI model. In dit hoofdstuk wordt uitgelegd wat het verschil is tussen deze hardware en wat geschikt is bij het trainen van een AI model.

5.2.2.1.1 CPU

Een CPU is er op gericht om taken serieel uit te voeren. Dat betekent dat een taak pas uitgevoerd wordt nadat een vorige taak volledig is afgerond. Daarbij komt dat een CPU goed is in het wisselen van de ene taak naar een compleet andere taak. Bij het trainen van een Deep Learning model worden er veel berekeningen gedaan. Als een CPU dit moet uitvoeren dan wordt elke berekening één voor één gedaan. Het effect hiervan bij het trainen van een AI model is dat het trainen lang zal duren. Daarom is een CPU niet geschikt voor het trainen van grote modellen.

5.2.2.1.2 GPU

Een GPU is erg geschikt voor het trainen van AI modellen. De reden hiervoor is dat een GPU meerdere gelijktijdige berekeningen kan uitvoeren (parallelle taken). Daarbij komt dat een GPU goed is in het verwerken van complexe berekeningen in kleinere subtaken. Deze subtaken kunnen parallel uitgevoerd worden (Inc, 2022). Bij Deep Learning is een grote dataset benodigd en zullen er veel berekeningen gedaan worden. Deze berekeningen kunnen met een GPU opgesplitst worden en gelijktijdig uitgevoerd worden.

5.2.2.1.3 TPU

Tenslotte is er nog een TPU, een afkorting voor: Tensor Processing Units. Een TPU is ontwikkeld door Google in 2016. Een TPU heeft een aantal eigenschappen waardoor het geoptimaliseerd kan werken voor Machine Learning modellen. Een voorbeeld hiervan is matrix vermenigvuldigingen. Het kan omgaan met grote vermenigvuldigingen, wat ingezet kan worden bij het trainen van een AI model (Sagar, 2022).

5.2.2.1.4 Uiteindelijke keuze

Wij hebben uiteindelijk besloten om gebruik te maken van een GPU bij het trainen van ons AI model. Een GPU is geschikt voor het trainen van AI modellen aangezien het complexe berekeningen parallel kan uitvoeren. Dit zorgt ervoor dat er sneller getraind kan worden dan een CPU. In Google Colab het alleen mogelijk is om gebruik te maken van een CPU en GPU. Een TPU is wel een optie in Google Colab maar wordt bijna niet toegewezen. Mocht er een TPU worden toegewezen dan is dit voor een korte periode.

5.2.2.2 Google Colab / Kaggle

Daarnaast hadden we de keuze tussen Google Colab en Kaggle. Dit waren allebei zogenaamde 'notebooks', waarbij je op een interactieve manier Python code kan combineren met documentatie, en dit allemaal op je browser (Google, z.d.). Het is erg geschikt voor Machine Learning, Data Analysis en educatie. En dus is het erg goed voor het maken van ons Proof of Concept. Iedere notebook had zijn eigen voor- en nadelen, daarom was het nogal lastig om een beslissing te maken welke notebook we wouden gebruiken. Toch hebben we een keuze gemaakt gebaseerd op de verschillen tussen de notebooks.

Het verschil tussen de notebooks was als volgt (Holmes, 2022):

Notebook:	Voordelen:	Nadelen:
Google Colab	<ul style="list-style-type: none"> + 12 uur limiet per dag + Makkelijk te combineren met Google Drive + Integratie met GitHub + TPU toegang + Overzichtelijke UI + Meest gebruikersvriendelijk voor nieuwe gebruikers 	<ul style="list-style-type: none"> - Je krijgt niet constant dezelfde hardware (soms beter / soms slechter) - Weinig schijfruimte - Upgrade is duur - Lastig om samen te werken (alles wordt overschreven) - Niet duidelijk hoeveel uur GPU tijd je krijgt.
Kaggle	<ul style="list-style-type: none"> + 30 uur limiet per week + Consistente hardware + Krachtigere GPU dan Colab 	<ul style="list-style-type: none"> - Veel eisen om Kaggle te gebruiken (bijvoorbeeld internet toegang na verificatie telefoon) - Geen TPU toegang

Omdat Google Colab meer voordelen had en wegens het feit dat wij al in een Google Drive werkten hebben wij gekozen voor Colab. In principe werkte deze notebook als verwacht, ons enige grote obstakel was samenwerken in een gezamenlijke Colab. Alle nieuwe code werd overschreden tijdens het typen en dus heeft iedereen in een apart bestand moeten werken.

Echter kwamen wij ook nieuwe obstakels tegen tijdens het individueel werken. Een van deze problemen was tijdens het trainen met veel epochs. Het duurde simpelweg te lang dat we het limiet hadden aangetikt voordat het afgerond was, of we raakte ons schrijf limiet aan. Zelf als we het trainen opsplitste in losse runs liepen we tegen problemen dat we bijvoorbeeld geen GPU toegewezen kregen.

Wegens dit probleem hebben we voor het trainen met grote hoeveelheden epochs gekozen om Kaggle te gebruiken. Het grootste verschil hiertussen was het limiet. Colab beperkte de gebruiker om dagelijks maar 12 uur te kunnen trainen, terwijl Kaggle 30 uur tot beschikking had per week. Deze 30 uur hebben we dus kunnen gebruiken toen de notebook overnacht getraind werd. Hierdoor heeft het model zich beter kunnen ontwikkelen en was het model meer accuraat.

Uiteindelijk kunnen we concluderen dat er geen specifieke notebook beter is, ieders hebben hun eigen voor- en nadelen en kunnen anders gebruikt worden. In ons geval zullen we Google Colab gebruiken voor kleinere taken zoals code voorbeelden en Kaggle voor het trainen van grote hoeveelheden data.

5.2.3 Transfer learning

Om het trainen van een model te versnellen kan je gebruik maken van een pre-trained model. Hierbij is een model getraind op een algemene dataset, waardoor het model sneller is in het oppakken van nieuwe informatie. Omdat deze models worden getraind op een algemene dataset is het nodig om hier nog extra data aan toe te voegen zodat het relevant wordt aan jouw situatie.

Het voordeel van transfer learning is dat een pre-trained model veel sneller is in het maken van voortgang. Hierdoor gaat het trainen veel sneller en heb je ook een kleinere dataset nodig.

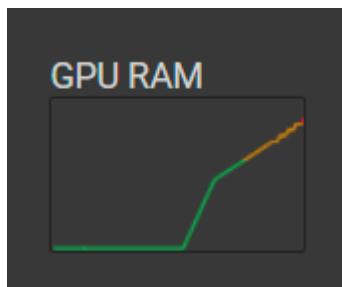
5.2.3.1 COCO

Voor transfer learning is het belangrijk om een model te gebruiken wat op een algemene manier is getraind. Een veel gebruikte dataset is COCO. COCO staat voor Common Objects In Context. Het is een dataset die meer dan 330000 afbeeldingen bevat en 80 verschillende categorieën. De dataset is gelabeld en kan dus worden gebruikt voor het trainen van je model. Een model wat met deze dataset is getraind maar niet de categorie bevat die je wilt gebruiken kan alsnog handig zijn. Het model zal dan sneller trainen dan als het model nog nooit objecten heeft gedetecteerd. Voor ons PoC hebben diverse modellen geprobeerd die de COCO dataset hebben gebruikt.

5.2.3.2 EfficientDet

Het eerste model dat wij gingen proberen was EfficientDet. Dit was een model dat was getraind op de COCO dataset. Bij EfficientDet kon je onze al gelabelde data importeren en dus leek het erg geschikt om te gebruiken. Echter liepen we al snel tegen meerdere obstakels aan zoals een memory leak.

De virtuele GPU kon het niet meer aan, omdat het niet genoeg RAM had. Ook stopte het model met trainen wegens een ‘out of memory’ error.

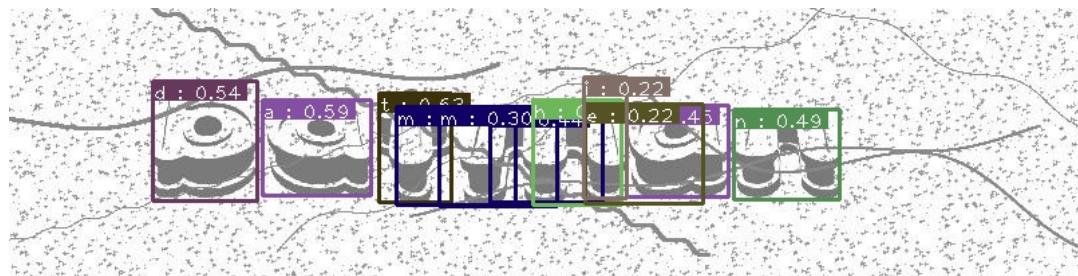


Figuur 7

Figuur 6

Oorspronkelijk dachten wij dat het probleem ontstond door de grote hoeveelheid epochs, maar zelfs bij een epoch hoeveelheid van maar 20 ging alles stuk. Alex heeft de code ook nog refactored, want er was een kleine kans dat er een fout in de code zat. Echter leverde ook [deze Colab](#) geen succes. Op Google leek het probleem al bekend te zijn bij diverse modellen van EfficientDet, toch hebben de oplossingen niets geholpen.

Uiteindelijk waren wij toch wel benieuwd naar het resultaat en hebben dus de afbeeldingen bekeken na een minimale hoeveelheid epochs.

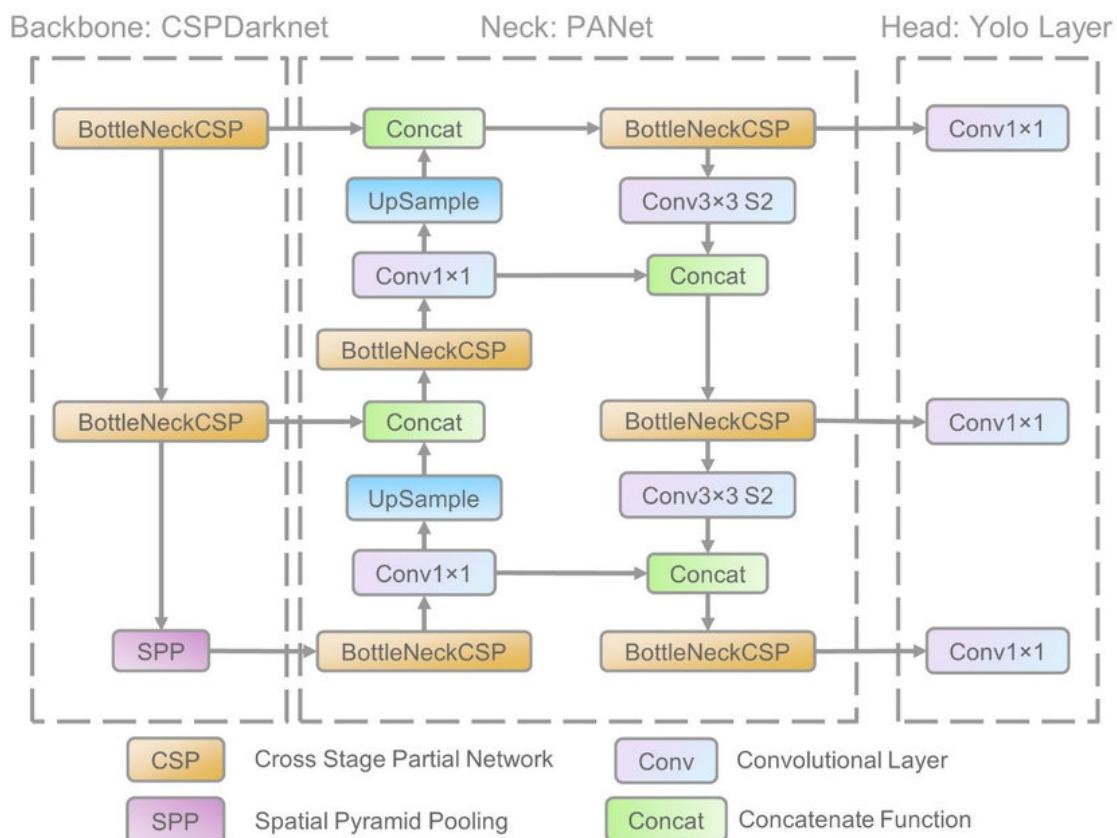


Figuur 8

Niet verrassend, maar het resultaat was erg slecht. Letters werden verkeerd, dubbel en/of met een erg lage accuraatheid gedetecteerd. En dus kan er concluderen dat EfficientDet niet geschikt is en niet meer gebruikt hoeft te worden.

5.2.3.3 YOLOv5

Het model waarmee wij de beste resultaten zagen was YOLOv5. YOLO staat voor You Only Look Once. Dit is een alternatief op de RCNN modellen. RCNN modellen zijn accuraat maar duren erg lang om te trainen omdat het detecteren uit meerdere stappen bestaat. Eerst wordt de regio gevonden waar de bounding box omheen wordt geplaatst, vervolgens vind de classificatie plaats en tot slot is er nog een verwerking op de output. Bij YOLO gebeurt dit allemaal in 1 stap. (Maindola, 2021)



Figuur 9 (Xu, 2021)

Het YOLOv5 model bestaat uit 3 onderdelen, een backbone, neck en head. De backbone is een CSPDarknet model. Bij deze stap wordt de input verwerkt en worden belangrijke features eruit gehaald. Vervolgens bij de neck wordt gebruik gemaakt van PANet. Hier wordt

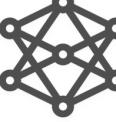
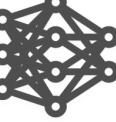
een feature pyramid gemaakt. Daarbij wordt hetzelfde object gedetecteerd op verschillende afbeelding formaten. Hierdoor wordt de accuraatheid van het model verbeterd.

In de laatste stap, de head, wordt de output voorbereid. De boxes, labels en probability score worden op de afbeelding geplaatst. (Rajput, 2020)

Het grote voordeel daarvan is dat het veel sneller is om te trainen. Verder is de accuraatheid ook erg goed. Inmiddels is dit de 5e versie van het YOLO model. Elke versie heeft verbeteringen met zich meegebracht in accuraatheid waardoor YOLOv5 vergelijkbaar en soms zelf beter is dan Faster RCNN. (Dwivedi, 2021)

5.2.3.3.1 Pre-trained checkpoints & Models

YOLOv5 biedt een aantal verschillende models en daarbij ook een pre-trained checkpoint die je kan gebruiken om gemakkelijk te starten.

				
Nano	Small	Medium	Large	XLarge
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
4 MB _{FP16} 6.3 ms _{V100} 28.4 mAP _{coco}	14 MB _{FP16} 6.4 ms _{V100} 37.2 mAP _{coco}	41 MB _{FP16} 8.2 ms _{V100} 45.2 mAP _{coco}	89 MB _{FP16} 10.1 ms _{V100} 48.8 mAP _{coco}	166 MB _{FP16} 12.1 ms _{V100} 50.7 mAP _{coco}

Figuur 10 (Jocher, 2022)

We hebben een test gedaan in Google Colab met een Tesla K80 GPU en kreeg daarmee de volgende voorspellingen van trainingstijd voor 120 epochs met 786 afbeeldingen van 640x640.

Model	Trainingstijd
YOLOv5n6	1.3 uur
YOLOv5s6	2 uur
YOLOv5m6	5 uur
YOLOv5l6	9 uur
YOLOv5x6	Te weinig GPU RAM.

We hebben de verschillende versies gebruikt en hebben gemerkt dat de nano variant erg snel trained maar niet een goede accuracy heeft. Deze is daarom meer geschikt om video's live te verwerken. Het Xlarge model daarentegen duurde erg lang om te trainen en gooide soms zelfs een VRAM Error afhankelijk van welke GPU je kreeg van Google Colab.

5.2.3.3.2 Hyperparameters

Hyperparameters zijn opties die je voorafgaand aan het trainen kan instellen. Deze parameters hebben effect op hoe het model traint. Het vinden van de juiste hyperparameters bij een specifiek probleem, kan alleen gevonden worden door het zelf uit te proberen of door te kijken naar eerdere uitwerkingen van anderen. Dit wordt ook bevestigd (Brownlee, 2019): "We cannot know the best value for a model hyperparameter on a given problem. We may use rules of thumb, copy values used on other problems, or search for the best value by trial and error." Het is ook mogelijk om automatisch te zoeken naar optimale hyperparameters. YOLOv5 heeft hier een feature voor genaamd hyperparameter evolution. Als dit is aangezet dat probeert het model zelf de optimale hyperparameters te vinden. Dit proces kan alleen erg lang duren. YOLOv5 zegt hier zelf over: "Note that evolution is generally expensive and time consuming, as the base scenario is trained hundreds of times, possibly requiring hundreds or thousands of GPU hours."

(Hyperparameter Evolution - YOLOv5 Documentation, z.d.)

Als hyperparameters gebruiken we de standaard opties alleen hebben we flip_lr op 0 gezet. Daarmee gaat het model er niet van uit dat een afbeelding kan worden gespiegeld. Voor letters dit belangrijk omdat bijvoorbeeld d en b niet kan worden gespiegeld. In onze testen bleek echter dat door deze aanpassingen het trainen ongeveer 2 keer zo lang duurde en het leek ook niet een verbetering door te voeren.

5.2.3.4 Faster R-CNN

Om een startpunt te hebben voor het werken met Faster R-CNN, zijn we begonnen met een voorbeeld [Google Colab van Roboflow](#). Volgens de instructie zou het mogelijk zijn om met een eigen dataset een model te trainen, en de voorbeeldafbeeldingen te testen. We hebben ervoor gezorgd dat alle stappen zijn doorlopen. In de configuratie zijn aanpassingen waar nodig gemaakt, zoals het importeren van de dataset en naamgeving van de geïmporteerde bestanden.

Het aspect dat opvalt bij het trainen met Faster R-CNN, is dat de tijdsduur langer in beslag neemt dan het model: YOLOv5. 1000 stappen duren ongeveer 15 minuten. In de beschrijving staat dat er zichtbaar resultaat is vanaf 200.000 stappen. Wat verder opvalt is dat de GPU capaciteit snel het limiet te bereikt met Faster R-CNN in Google Colab. Dit zorgt ervoor dat er waarschuwingen getoond worden bij het trainen, en het trainingsproces uiteindelijk stopt. Het hoge GPU verbruik is te zien in figuur 11. Na een aanpassing in de 'batch size' is het GPU verbruik verminderd. In figuur 12 is het effect bij het aanpassen van de 'batch size' in te zien.

Figuur 11

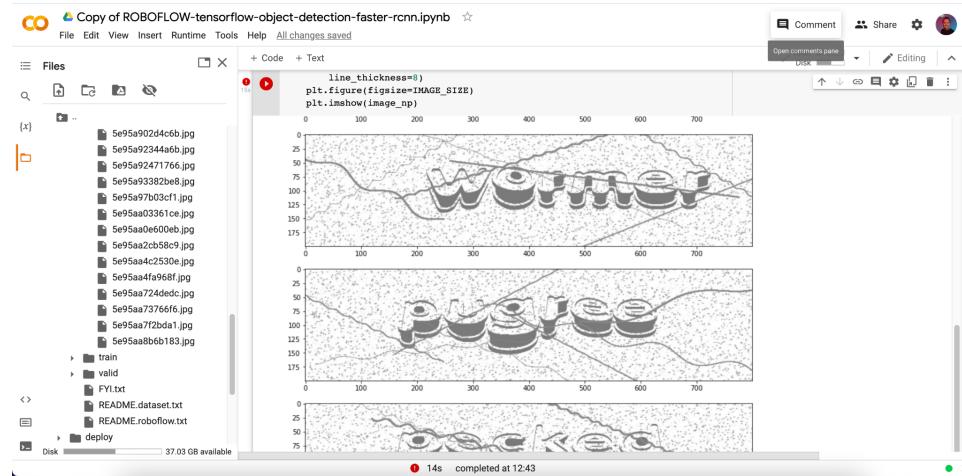


Figuur 12



Bij het beginnen met trainen zijn het aantal stappen ingesteld op 2000. Nadat het trainingsproces afgelopen was, zijn we verder gegaan met herkennen van

voorbeeld afbeeldingen. Hieruit werd geen enkele afbeelding herkend. Het aantal stappen zijn daarna verhoogd tot uiteindelijk 5000 stappen. Bij meer stappen werd het trainingsproces onderbroken door Google Colab, en kon er niet verder getraind meer worden. Na dit trainingsproces werd geen enkele afbeelding herkend. Omdat het werken met Faster R-CNN geen resultaten op heeft geleverd, zijn we verder gegaan met het trainen van het Yolov5 model. De snelheid van Yolov5 ten opzichte van Faster R-CNN wordt ook bevestigd (Dwivedi, z.d.): “Finally, comparing the two models, it can be seen that YOLOv5 has obvious advantages in operating speed. The small YOLOv5 model runs faster by about 2.5 times and has better performance when detecting smaller targets.”



Figuur 13

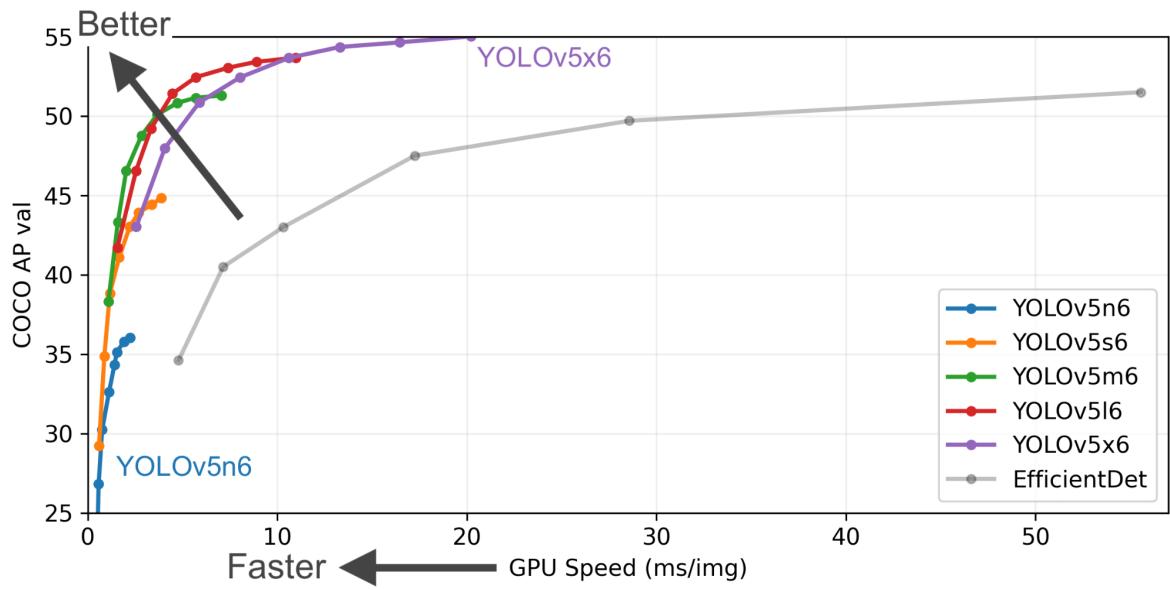
5.2.3.5 EfficientDet (TensorFlow)

We hadden al een eerdere test gedaan met EfficientDet, maar daar hadden we een issue mee waardoor de GPU Ram de hele tijd vol liep. Wat later in het project hebben we dit model een tweede kans gegeven met een andere implementatie in [deze Colab](#), nu werkte het model wel.

Wat ons opviel in vergelijking met YOLOv5 was dat het model erg groot was en daardoor veel GPU Ram vereiste. We konden daardoor alleen het kleinste model gebruiken en ook nog met een verminderde Batch size van 8. Het trainen zelf duurde mede hierdoor een stuk langer.

We hebben gekeken naar bestaande onderzoeken om te bepalen of het waard is om te wisselen naar EfficientDet. Wat we vonden was dat het trainen van het model ongeveer 6x langer duurt in vergelijking met YOLOv5 terwijl de performance vergelijkbaar was. (An experiment with YOLOv3-4-5 and EfficientDet for infrastructure asset management – Result! Data, 2020)

Het bleek dus dat onze bevindingen overeenkomen met wat andere mensen ook hadden ervaren en daarom hebben we besloten om door te blijven werken met YOLOv5.



Figuur 14 (GitHub - Ultralytics/Yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite, z.d.)

5.2.3.6 Mask R-CNN

R-CNN is erg geschikt voor het oplossen van image segmentation, dit is de taak om diverse objecten te onderscheiden op een afbeelding. Daarbij maakt deze Mask variant gebruik van een zogenaamde instance segmentation, dat houdt in dat ze verschillende instanties van dezelfde objecten toegewezen worden bij verschillende labels.

In [deze Colab](#) wordt er een afbeelding geladen met verschillende COCO annotaties. Daarna worden de voorspellingen geladen, deze waren al eerder gedaan en dus was het hele process een beetje nagespeeld. Toch was het uiteindelijk resultaat best indrukwekkend, want er was een duidelijk overzicht te tonen met het object en de zekerheid. Alles was met een redelijk hoge accuraatheid en je kon zelfs de data filteren op de hoeveelheid objecten & minimale zekerheid.



Figuur 15

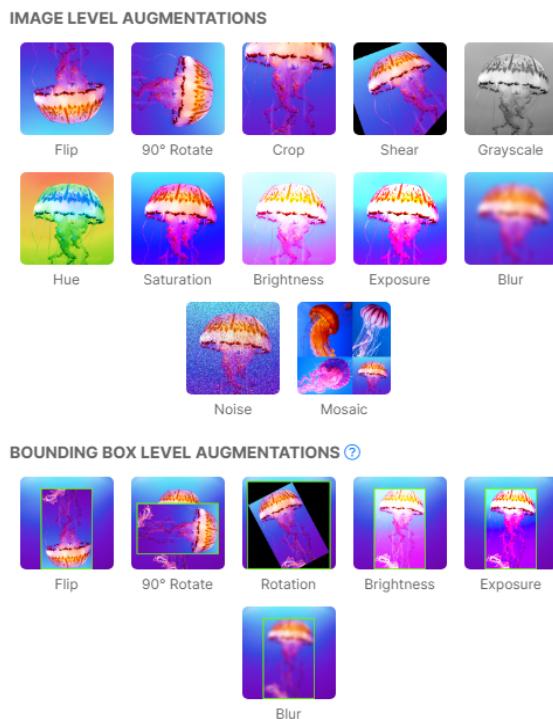
5.3 Welke data hebben we nodig om het model te kunnen trainen?

Om de accuraatheid van het model te verhogen hebben we zo veel mogelijk afbeeldingen nodig van de 3D CAPTCHA set. Verder is het belangrijk dat de afbeeldingen een hoge resolutie hebben en dat er meerdere variaties van letters in de afbeeldingen zitten. Het is hierbij belangrijk om rekening te houden met de verdeling van de classes. We hebben bijvoorbeeld veel woorden waar de letters a of e in zitten. Dit zorgt er voor dat er een bias is voor het model om a of e als antwoord te geven.

5.3.1 Data Augmentation

Data Augmentation is het automatisch aanpassen van afbeeldingen om zo je dataset te vergroten door meer varianten te creëren en/of de afbeelding(en) geschikter te maken voor je model (Saxena, 2021). Wegens het feit dat je meer data hebt om je model mee te trainen, kan het model ook accurater worden.

In ons geval hebben wij Roboflow (Roboflow: Give Your Software the Power to See Objects in Images and Video, z.d.) gebruikt om Data Augmentation toe te passen. Deze tool heeft diverse standaard instellingen die toegepast kunnen worden op de bestaande afbeeldingen. Echter wisten wij nog niet welke van deze opties het meest geschikt was voor ons model en dus hebben wij diverse opties getest. Onze verwachting hierbij was dat doordat we meer afbeeldingen hadden dat de accuraatheid omhoog ging. We hebben dit getest door eerst een baseline meting te maken zonder augmentations en deze te vergelijken met een model wat met augmentations was getraind. Het resultaat van dit onderzoek zal hieronder verteld worden.



Figuur 16 (Roboflow, z.d.)

5.3.1.1 Verschillende soorten data augmentation

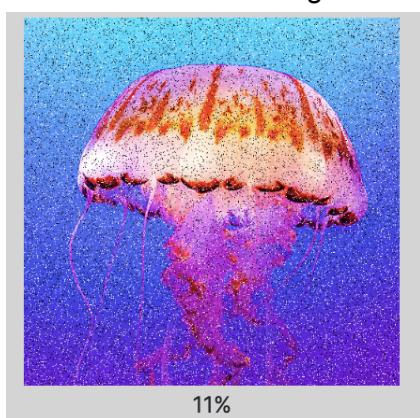
Zoals al verteld werd zijn er verschillende soorten Data Augmentation opties die toegepast kunnen worden op een dataset. Elke soort data augmentation heeft een specifieke eigenschap en kan zorgen voor een bepaald effect bij het trainen. In Roboflow is het mogelijk om de sterkte in te stellen bij een data augmentation optie. Door deze optie is het mogelijk om een data augmentation optie te versterken of te verzwakken.

Roboflow biedt veel opties, maar voor onze usecase vallen er al veel af. Het aanpassen van kleuren heeft geen zin omdat de CAPTCHA's zwart-wit zijn. Hierdoor vallen greyscale, hue, saturation, brightness, exposure al af. Verder zijn alle afbeeldingen scherp dus heeft blur ook geen zin. flip en rotate zijn ook geen opties omdat letter een andere betekenis kunnen hebben met die augmentation bijvoorbeeld bij d, b, q en p. Mosaic hebben we niet bekeken. Dan blijven er 4 mogelijke augmentations over.

5.3.1.1.1 Noise

Bij de data augmentation: "Noise", wordt er ruis toegevoegd aan een afbeelding. Dit zorgt ervoor dat de dataset achtergrondruis krijgt. Het toevoegen van ruis zorgt ervoor dat "overfitting" voorkomen kan worden. Bij het gebruiken van deze data augmentation, neemt de "entropie" toe. De betekenis hiervan is het toevoegen van de mate van willekeurigheid. Zoals bevestigd wordt (Entropie: wat is dat precies? - Mr. Chadd Academy, z.d.): "Het is een maat voor de hoeveelheid chaos in een systeem en geeft aan hoe deeltjes verdeeld zijn in een systeem." De afbeelding blijft hierdoor niet de originele staat en er ontstaat willekeurigheid. We moeten er op letten dat we niet te veel noise toevoegen aan de afbeeldingen. Door noise toe te voegen verlaagt het informatiegehalte in de afbeelding waardoor het voor het model moeilijker is om de letters te herkennen.

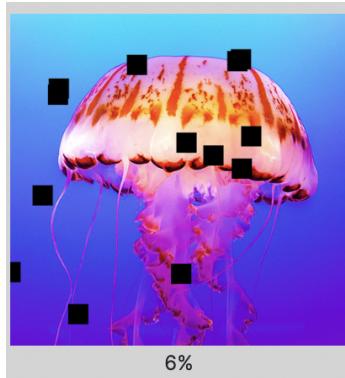
Zoals (Brownlee, 2019) uitlegt: "Adding noise means that the network is less able to memorize training samples because they are changing all of the time.". Doordat er steeds willekeurige ruis toegevoegd, is elke afbeelding uniek. Het model heeft daardoor meer moeite om de letters te kunnen herkennen. Op deze manier wordt het model robuuster. Dit wordt ook bevestigd (Brownlee, 2019): The addition of noise during the training of a neural network model has a regularization effect and, in turn, improves the robustness of the model.



Figuur 17 (Roboflow, z.d.)

5.3.1.1.2 Cutout

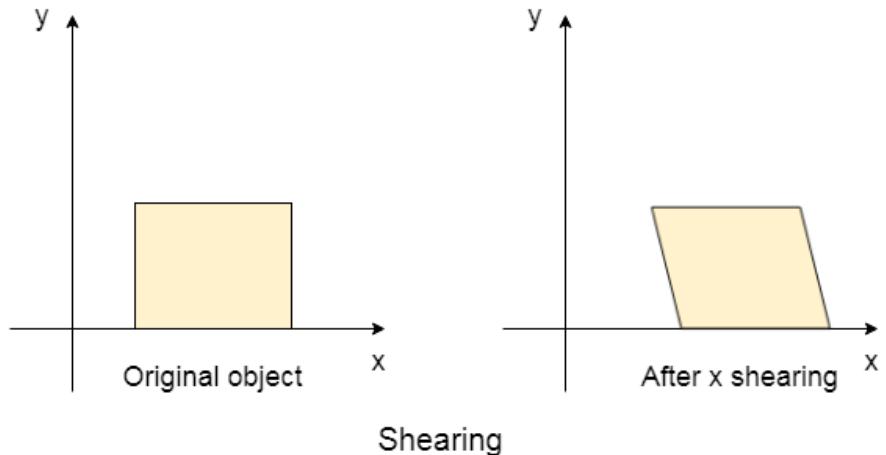
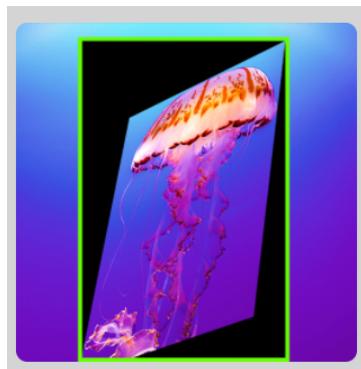
Verder is er de data augmentation: "Cutout". Er wordt hierbij willekeurige vlakken toegevoegd over de afbeelding heen. Dit zorgt ervoor dat het model robuuster wordt zoals bevestigd wordt (DeVries Et Al., 2017): "Cutout is an image augmentation and regularization technique that randomly masks out square regions of input during training and can be used to improve the robustness and overall performance of convolutional neural networks". De reden waarom "Cutout" gebruikt wordt is dat het model beter leert waar het op moet focussen (DeVries Et Al., 2017): "we not only better prepare the model for encounters with occlusions in the real world, but the model also learns to take more of the image context into consideration when making decisions".



Figuur 18 (Roboflow, z.d.)

5.3.1.1.3 Shear

Bij de data augmentation: "Shear" worden 2 hoeken van de afbeelding verplaatst om zo de hoek van de gehele afbeelding te veranderen. Dit is een manier om de data aan te passen zonder hem te draaien en biedt dus een extra variatie aan de dataset.

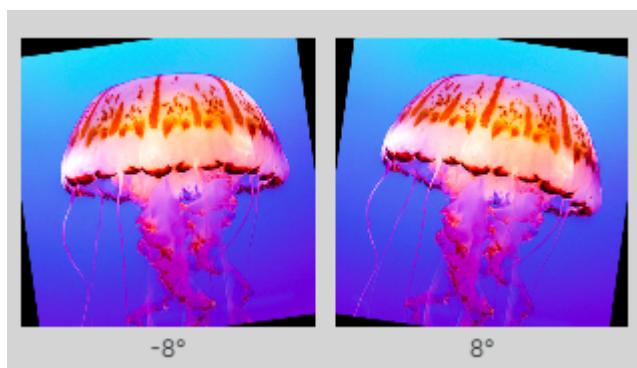


Figuur 19 (Roboflow, z.d.)

Figuur 20 (JavaFX Shearing - Javatpoint, z.d.)

5.3.1.1.4 Rotate

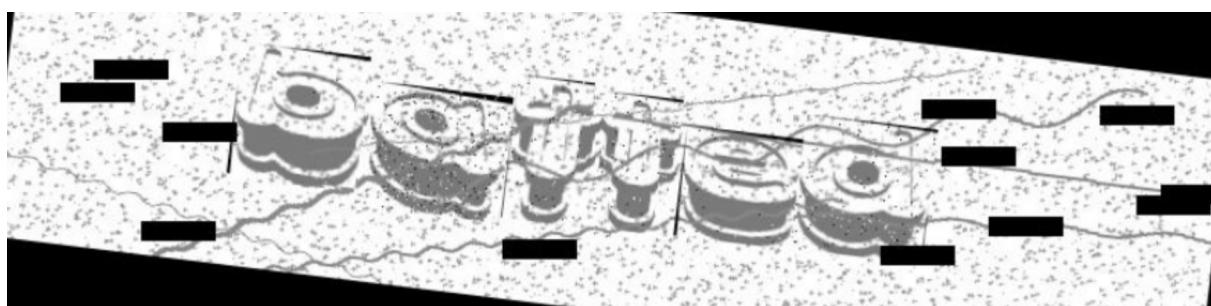
Bij de data augmentation: “Rotate” wordt een afbeelding met de klok mee of tegen de klok in gekanteld. Het gebruik van de data augmentation: “Rotate”, kan helpen bij “overfitting” door middel van de variatie in perspectieven van een afbeelding. Dit wordt ook bevestigd (Solawetz, 2020): “Random rotation can also help combat potential overfitting. Even in cases where the camera position is fixed relative to the subjects your model is encountering, random rotation can increase variation to prevent a model from memorizing your training data.”. In sommige gevallen is het niet effectief om rotatie op een dataset toe te passen. Dit is het geval als er belangrijke informatie staat in de hoeken van een afbeelding. Dit wordt ook bevestigd (Solawetz, 2020): “Thus, the first reason you should consider not using random rotation is if there is valuable content in the original corners of your images.”



Figuur 21 (Roboflow, z.d.)

5.3.1.2 Toepassen van data augmentation

In ons geval bevat onze huidige dataset 404 afbeeldingen, echter door gebruik te maken van data augmentation kunnen we tot wel 3x zo veel afbeeldingen komen. We hebben er dus voor gekozen om gebruik te maken van de rotate, shear, noise en cutout opties. Als we alle augmentations toepassen ziet de input er zo uit:



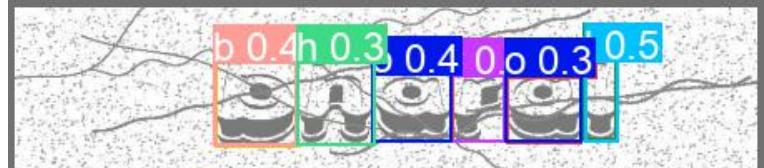
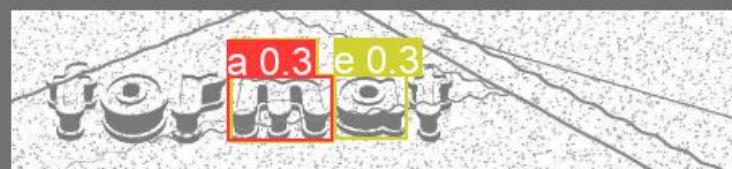
Figuur 22 (Roboflow, z.d.)



Figuur 23 (Roboflow, z.d.)

De reden dat wij augmentation hebben proberen toegepast is omdat we merkte dat bijna elke letter gezien werd als een 'e' (zie onderstaande afbeelding 24/25). Hierin is dus een duidelijke bias te zien naar deze specifieke letter. Dit komt omdat er in de dataset te veel afbeeldingen zijn met de letter 'e', in verhouding tot de andere letters.

Om het verschil te demonstreren hebben we een test gedaan met het kleinste model voor 60 epochs. Dit is erg weinig tijd voor het model en dit zien we ook in de resultaten maar het toont wel aan dat data augmentation een groot verschil heeft gemaakt. Het model wat is getraind zonder data augmentation heeft veel van de letters niet gelabeld, sommige van de letters die zijn gelabeld zijn incorrect gelabeld.

Zonder augmentation:	Met augmentation:
 strain.jpg rr 12b2910617128de540534bd4fc	 strain.jpg rr 12b2910617128de540534bd4fc
 format.jpg rr d5cd0460251fed1b9a0a45554	 format.jpg rr d5cd0460251fed1b9a0a45554
 filled.jpg rr 04c7729eb0f81b0536142ecbb	 filled.jpg rr 04c7729eb0f81b0536142ecbb

Figuur 24

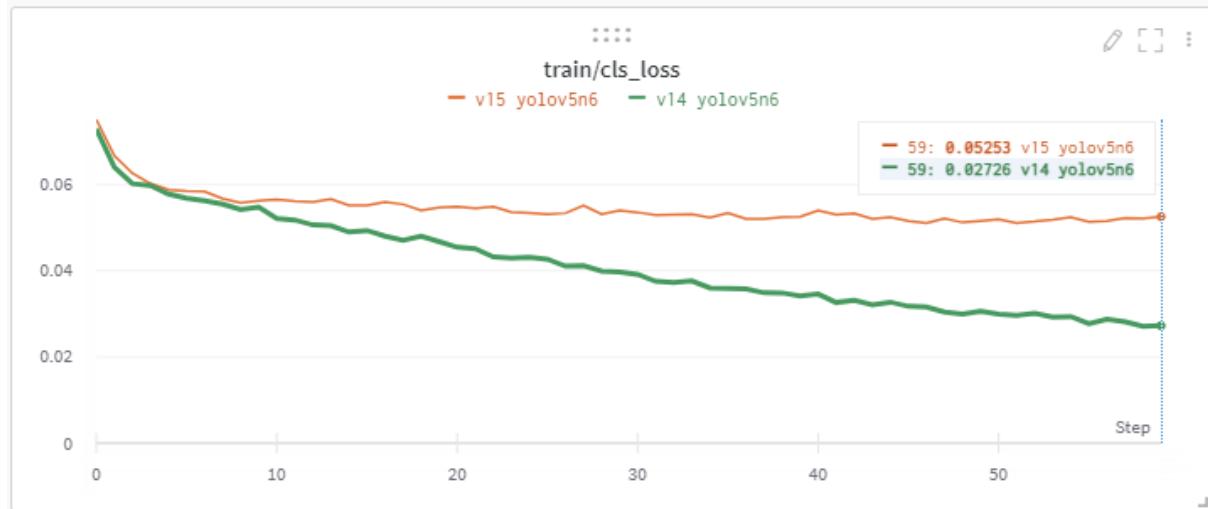
Figuur 25

Bij de dataset met data augmentation zie je al een flinke verbetering. Alle letters zijn gelabeld, veel van de letters lijken ook te kloppen, maar de confidence is wel erg laag. Om dit te verbeteren zou je langer kunnen trainen. Om te kijken of dit effect zou hebben kunnen we kijken naar de 'cls_loss function'.

Zie deze Google Colab bestand voor een demo:

<https://colab.research.google.com/drive/1Fi0Zr2zo4KwBF069jD6PNsAJyXFU-xb7?usp=sharing>

5.3.1.3 Cls_loss vergelijking



De groene lijn is het model wat data augmentation toepast.

Figuur 26 (Wandb.ai, z.d.)

De zogenaamde ‘cls_loss’ toont hoe accuraat het model is in het bepalen van de class. Dus of een ‘e’ als ‘e’ of een andere letter wordt gezien. In dit geval is lager beter omdat dat betekent dat het minder vaak mis gaat. Je ziet dat de oranje lijn horizontaal gaat, dit betekend dat het model niet meer kan leren van de dataset die wordt gebruikt. De groene lijn met de grotere dataset door data augmentation daarentegen is zelfs op stap 60 nog niet horizontaal aan het gaan. Dat betekend dat het model zelfs nog beter zou kunnen worden als er langer wordt getraind. Hieruit kunnen we tot nu toe concluderen dat we een grote dataset nodig hebben en dat data augmentation een goede manier is om hier aan te komen.

5.4 Hoe zeker moet het model zijn over zijn keuze om aan te geven welke letter correct is?

Bij het lezen van een CAPTCHA door middel van AI moeten wij niet kijken naar het gehele woord, maar naar de individuele letters waaruit het woord bestaat.

Het is erg lastig om te interpreteren hoe zeker een model exact moet zijn om aan te geven of een letter correct is. Toch zijn er verschillende manieren om een indicatie te kunnen geven.

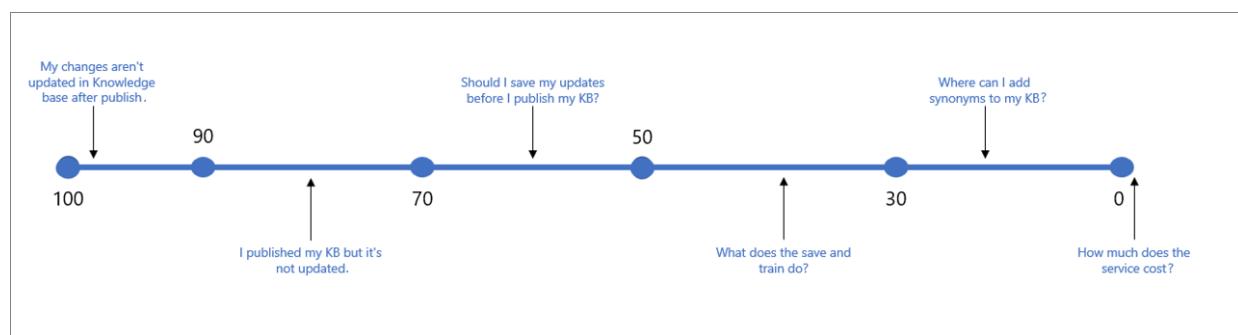
5.4.1 Confidence score

Een manier waarvan je het kan bekijken is simpelweg naar de hoeveelheid letters. In het alfabet zitten 26 letters. Dit zou dus betekenen dat een model '1/26' zeker moet zijn, want dat zijn de totaal aantal verschillende letters die voor kunnen komen. Dit is voor onze situatie alleen niet zeker genoeg. Voor het raden van een letter moet er namelijk een duidelijke uitkomst uitkomen van 1 letter. Daarnaast wanneer we in de toekomst eventueel onze dataset willen uitbreiden met afbeeldingen die hoofdletters bevatten gaat deze minimale zekerheid al naar 1/52 en dat is erg laag. Daarom moeten we zoeken naar een alternatief.

Deze alternatief was om de resultaten van een model te bekijken gebaseerd op zogenaamde 'confidence scores'. Deze confidence is dan de waarde hoe zeker het model is over een voorspelling. Echter is er niet een bepaalde confidence score die correct is. Toch zijn er wel richtlijnen die je een idee kunnen geven. De drie meest voorkomende confidence score types zijn:

- 0 tot 1: Hier is het makkelijk voor de gebruiker om het te begrijpen, alleen hoe zeker is 1 nou? Het hoeft niet te betekenen dat het correct is.
- $-\infty$ tot $+\infty$: Hierbij kun je bijna altijd de twee confidence scores vergelijken, echter zegt dit niet veel voor de gebruiker zelf.
- Expressions (low, medium, high): Makkelijk te begrijpen, maar niet in wiskundige functies te benutten

En Microsoft heeft bijvoorbeeld de volgende richtlijnen bedacht waarbij ze aangeven waar een confidence score voor staat (Microsoft Docs, 2021). Meer dan 0.7 is een goede kans dat het antwoord klopt. Tussen 0.3 en 0.7 zal gedeeltelijk kloppen en minder dan 0.3 is waarschijnlijk een slechte optie.



Figuur 27 (Microsoft Docs, 2021)

5.4.2 Precision en Recall

Ook de ‘precision’ is van belang. Dit is de verhouding tussen alle correcte voorspellingen en alle voorspellingen die als ‘correct’ teruggegeven worden, maar niet daadwerkelijk correct zijn (Grandperrin, 2021). Dit verschil noemen we false & true positives. Hiervan heb je ook negative varianten van. Deze positive & negatives gaan als volgt:

- True positive: Voorspelling ‘true’ en is correct
- True negative: Voorspelling ‘false’ en is correct
- False positive: Voorspelling ‘true’ en is fout (daadwerkelijke antwoord was ‘false’)
- False negative: Voorspelling ‘false’ en is fout (daadwerkelijke antwoord was ‘true’)

Een voorbeeld hiervan is een model wat auto’s probeert te herkennen op basis van een foto. Je hebt bijvoorbeeld een foto met 10 mensen en 12 auto’s. Het model identificeert 8 auto’s. Van deze 8 zijn er maar 5 daadwerkelijk auto’s (True positive). De andere 3 waren mensen (False positive). 7 auto’s zijn gemist (False negative) en 7 mensen waren correct niet gezien als mens (True negative).

De berekening om precision uit te rekenen is als volgt:

$$\text{Precision} = \frac{tp}{tp + fp}$$

Figuur 28 (Google, z.d.-a)

Hierbij delen wij de true positives door de true positives + false positives (alle gegeven positives, zowel correcte als incorrecte).

Ten slotte is ook ‘recall’ belangrijk. Met recall bereken je welk deel van alle gelabelde objecten correct voorspeld is. Bij een hoge recall heb je weinig false negatieve en dus heeft het model weinig onterecht aangemerkt als zijnde een object. Een lage recall wil zeggen dat je model erg veel objecten mist. De recall kun je ook berekenen aan de hand van de verhouding tussen true en false positives en negatives.

Dat ziet er als volgt uit:

$$\text{Recall} = \frac{tp}{tp + fn}$$

Figuur 29 (Google, z.d.-a)

Er vanuit gaande dat je een confidence score hebt van 0 tot 1, is het voor een mens logisch om alles boven de 0,5 voor waar aan te nemen. De zogenaamde ‘threshold value’ is in dit geval 0,5, oftewel de minimale confidence score waarbij een voorspelling als waar wordt aangenomen.

Je kunt de threshold aanpassen om het algoritme beter aan je eisen aan te laten sluiten. Als het bijvoorbeeld belangrijk is dat zo veel mogelijk voorspellingen die het model doet juist zijn, en je dus een hoge precision wilt hebben, neem je een hoge threshold. Als een hoge recall belangrijk is, neem je een lagere threshold.

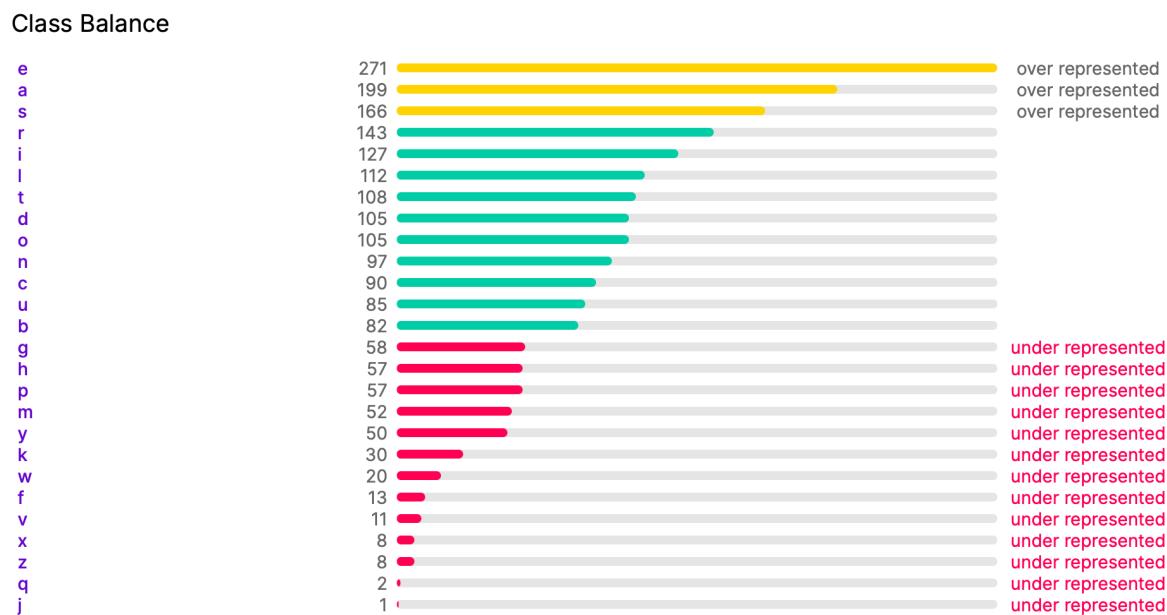
Precision en recall zijn dus belangrijk om confidence scores goed te kunnen interpreteren.

5.5 Hoe kunnen we de accuraatheid van het model verbeteren?

We maken gebruik van verschillende tools en statistieken om de accuraatheid van het model te verbeteren. Aan de hand van de informatie die we verkrijgen uit de tools die we gebruiken, maken we aanpassingen aan het model. Op die manier komen we te weten wat de winst in accuraatheid is bij de verschillende aanpassingen die we aan het model doen.

5.5.1 Balans in dataset

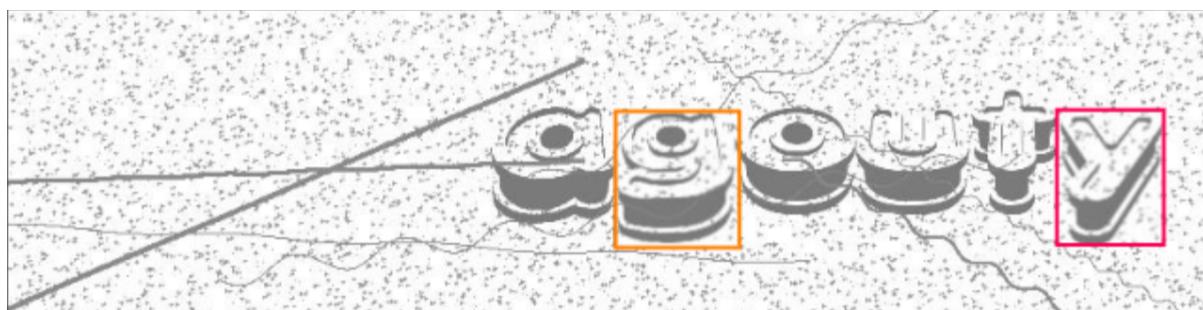
In RoboFlow, de tool waar we onze data mee gelabeld hebben, kun je onder het kopje 'Health check' zien welke class (in ons geval letter), hoe vaak voorkomt in de data. (Dataset Health Check - Roboflow, z.d.) Deze verdeling is te zien in onderstaande afbeelding (Figuur 30).



Figuur 30 (Wandb.ai, z.d.)

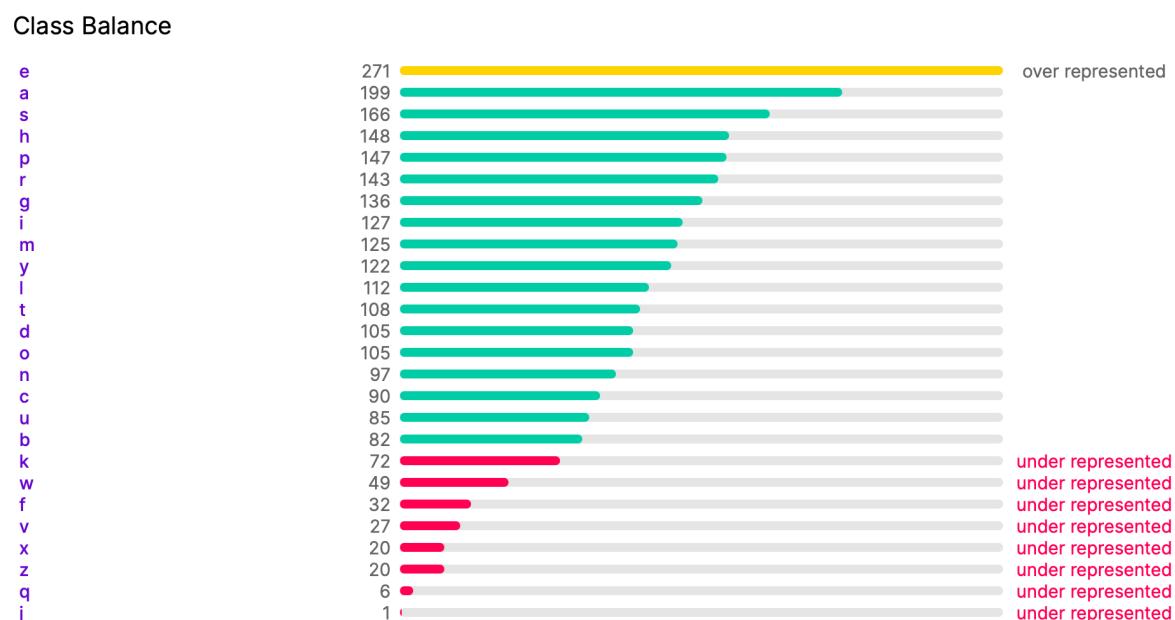
Hier is te zien dat een aantal letters (e, a, en s) oververtegenwoordigd zijn en een aantal letters (g, h, p, m, y, k, w, f, v, x, x, q, j) ondervertegenwoordigd zijn. Er is een onbalans aanwezig in de data. Deze onbalans kan zorgen voor bias. Dat wil zeggen dat het model letters vaker zal herkennen als een letter die vaak in de data voorkomt en dat het model letters die niet vaak in de data voorkomen vaak onjuist zal herkennen. Om dit effect te verkleinen is het handig om de data meer in balans te brengen. (Brems, 2020)

Tijdens het experimenteren met Roboflow kwamen we erachter dat je classes uit kan sluiten van een dataset. We hebben in Roboflow een nieuwe versie van de dataset gemaakt. Hierbij hebben we alle letters die niet ondervertegenwoordigd zijn uitgesloten. Deze dataset bevat dus alleen de letters die ondervertegenwoordigd zijn. De andere letters worden in deze dataset beschouwd als niet gelabeld, zoals te zien in het plaatje hieronder. Alleen de letters met een vakje eromheen, in dit voorbeeld de 'g' en de 'y', zijn gelabeld.



Figuur 31

Deze dataset hebben we vervolgens samengevoegd met de originele dataset. Het resultaat hiervan is een dataset die meer in balans is.



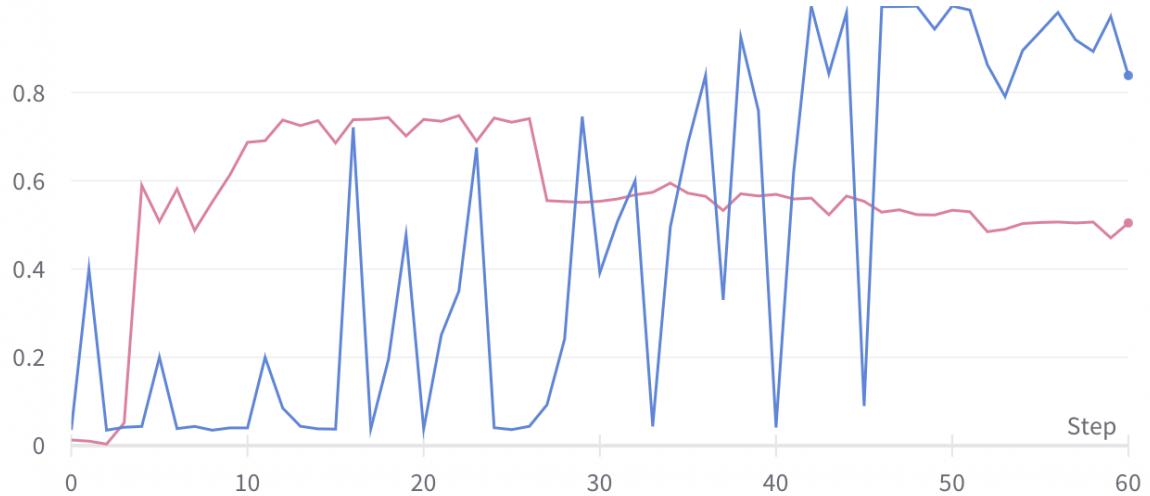
Figuur 32 (Wandb.ai, z.d.)

We hebben het YOLOv5 model getraind met de originele dataset en de nieuwe dataset en gekoppeld aan Weights & Biases. Weights & Biases is geïntegreerd in YOLOv5. (YOLOv5 - Documentation, z.d.) Weights & Biases is een platform om sneller betere modellen te maken. Een van de functies van dit platform is het visualiseren van resultaten.(Weights & Biases - Documentation, z.d.)

De door Weights & Biases geproduceerde grafieken kunnen we gebruiken om de resultaten van het model, getraind met de originele dataset, te vergelijken met die van het model die getraind is met de meer gebalanceerde dataset.

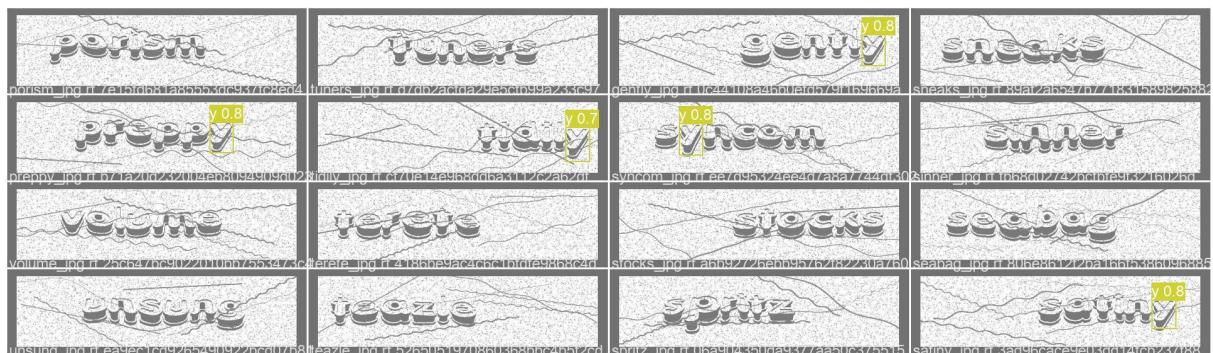
5.5.2 Precision

Zoals eerder werd verteld is precision de verhouding tussen alle correcte voorspellingen en alle voorspellingen die als 'correct' teruggegeven worden, maar niet daadwerkelijk correct zijn (Grandperrin, 2021). In het geval van CAPTCHA's kun je dat zien als de verhouding tussen alle letters die het model gevonden heeft en correct voorspeld heeft, en alle gelabelde letters. In onderstaande grafiek (Figuur 33) wordt de precision gedurende het trainingsproces weergegeven. De roze lijn is hierbij de originele dataset en de blauwe lijn de beter gebalanceerde dataset.



Figuur 33 (Wandb.ai, z.d.)

Het valt meteen op dat de grafiek van de beter gebalanceerde dataset hogere pieken en dalen laat zien dan de grafiek van de originele dataset. Het ene moment is de precision erg hoog en het andere moment erg laag. Een hoge precision wil zeggen dat de voorspellingen die het model doet vaak juist zijn. Als het model weinig voorspellingen doet, heb je ook maar weinig goede voorspellingen nodig om tot een hoge precision te komen. Je hebt dan ook maar weinig foute voorspellingen nodig om tot een lage precision te komen. Een mogelijke verklaring van de grote pieken en dalen is dan ook het lage aantal voorspellingen dat het model doet. Dit blijkt ook uit de afbeeldingen van de voorspellingen die het model heeft gedaan. In onderstaande afbeeldingen (Figuur 34) zijn een aantal voorspellingen te zien die het model gedaan heeft. Dat zijn er slechts vier.



Figuur 34 (Wandb.ai, z.d.)

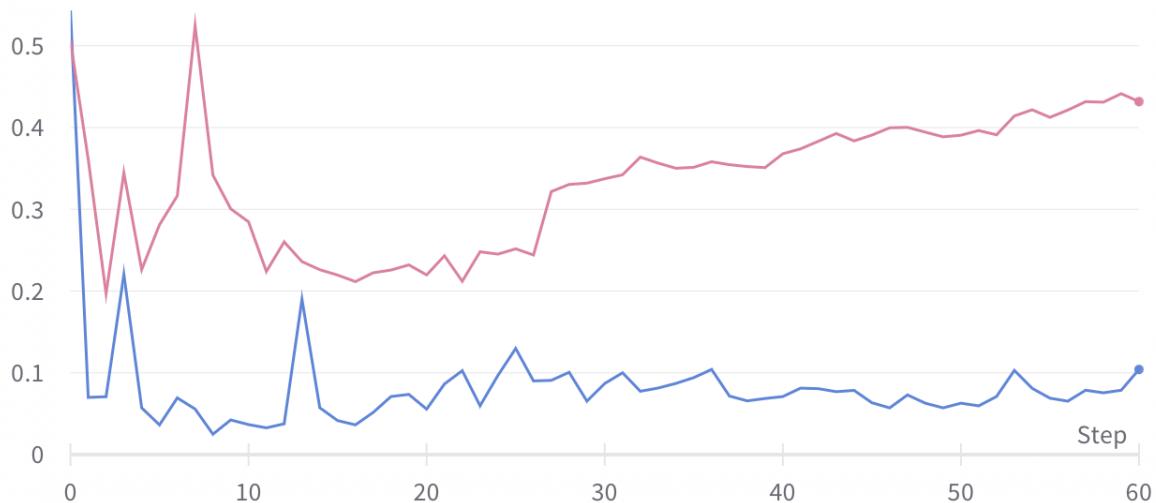
Dat zijn er erg weinig als je dat vergelijkt met het aantal voorspellingen dat het model, getraind met de originele dataset, doet. Zie onderstaande afbeelding. (Figuur 35)



Figuur 35 (Wandb.ai, z.d.)

5.5.3 Recall

In onderstaande afbeelding (Figuur 36) zie je de recall grafiek van het YOLOv5 model, getraind met de originele dataset (de roze lijn) en getraind met de gebalanceerde dataset (de blauwe lijn).

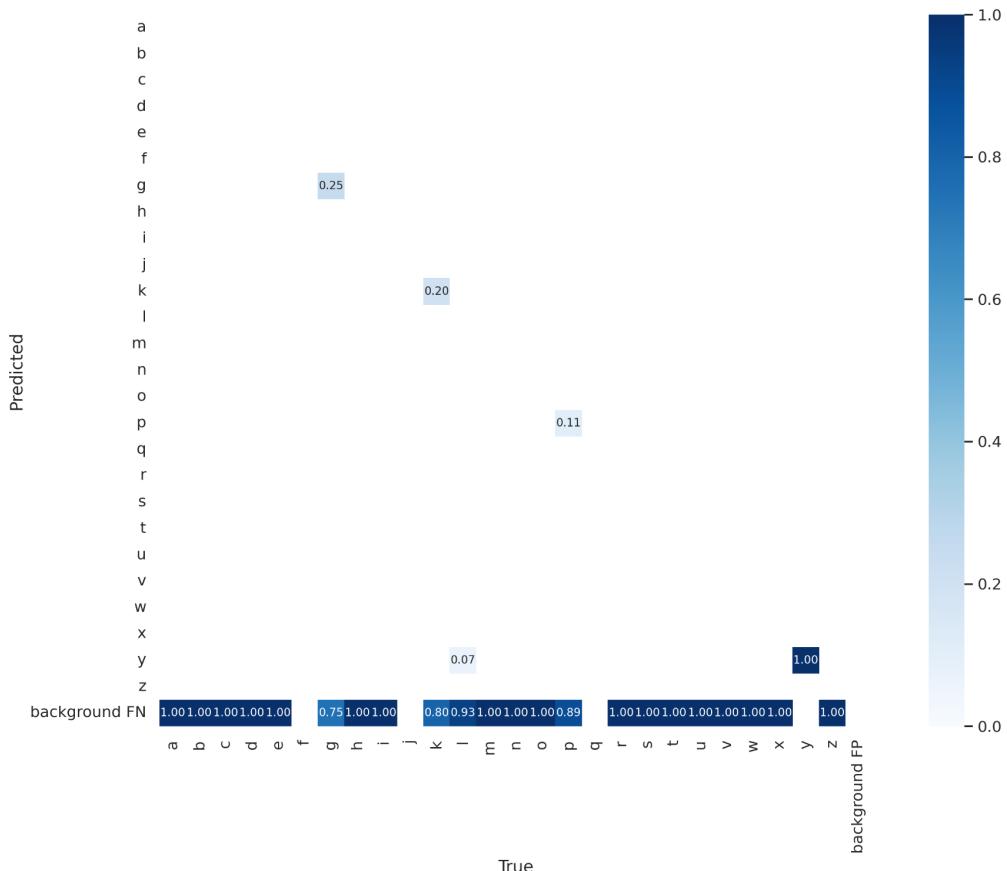


Figuur 36 (Wandb.ai, z.d.)

Zoals te zien in de grafiek is de recall van de gebalanceerde dataset een stuk lager dan die van de originele dataset. Een lage recall wil zeggen dat er veel letters door het model zijn gemist. Dit ligt in lijn met de eerder genoemde verklaring van de pieken en dalen in de precision grafiek.

In afbeelding 37 zie je welke letters het model, getraind met de gebalanceerde dataset, voorspelt dat ze zijn. De afbeelding is te lezen als een soort tabel. Op beide assen staan alle classes die de data bevat. De getallen staan voor welk deel van een class x herkend wordt als class y. Hierbij staan de werkelijke classes op de x-as en de classes zoals het model ze herkend op de y-as. Er is te zien dat er zeer weinig letters door het model wordt geclasseerd als zijnde een letter. Oftewel, er worden heel veel letters gemist. Dit komt overeen met wat we gezien hebben in de precision en recall grafiek. Veruit de meeste letters worden door het model herkend als achtergrond. Dit heeft waarschijnlijk te maken met hoe de dataset is opgebouwd. Bij veel afbeeldingen zijn enkel de ondervertegenwoordigde letters gelabeld. Het model denkt daardoor dat de niet gelabelde letters geen letters zijn,

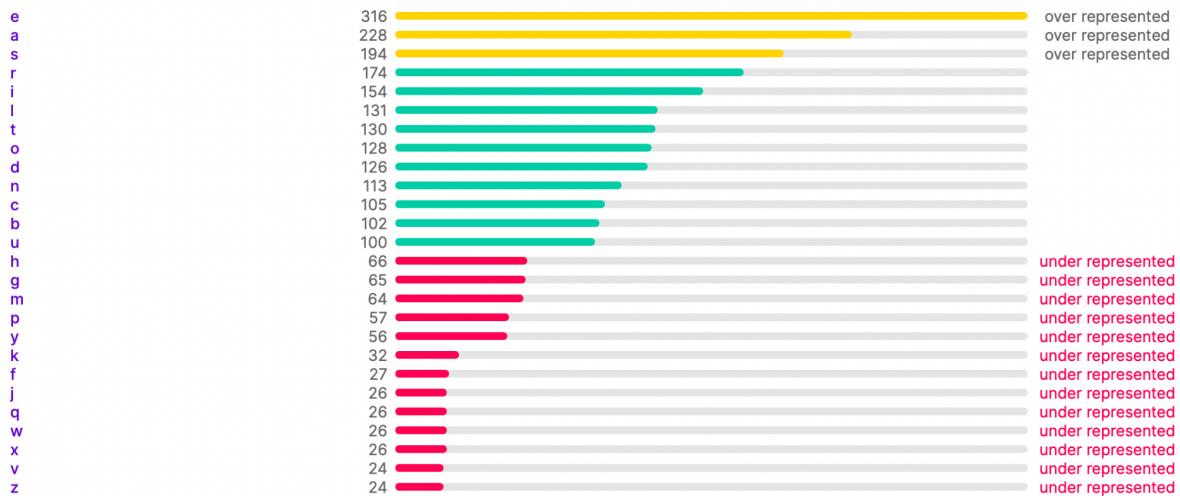
maar achtergrond is. De manier waarop we de dataset gebalanceerde hebben gemaakt lijkt dus de oorzaak te zijn van het weinig aantal letters dat herkend wordt.



Figuur 37 (Wandb.ai, z.d.)

Het uitsluiten van classes is geen goede oplossing gebleken om het model beter te laten presteren. Een andere mogelijke oplossing is om extra afbeeldingen te maken met letters die niet vaak voorkomen in de originele dataset. We hebben letters die het minst vaak voorkomen uit de originele afbeeldingen gehaald en daar nieuwe afbeeldingen van gemaakt. Deze hebben we vervolgens toegevoegd aan de originele dataset en gelabeld. Dit levert een dataset op waarbij elke letter in iedere geval 12 keer vertegenwoordigd is. De situatie is nu zo dat alle letters in de afbeeldingen gelabeld zijn. Dit zou moeten voorkomen dat het model de letters als achtergrond gaat herkennen.

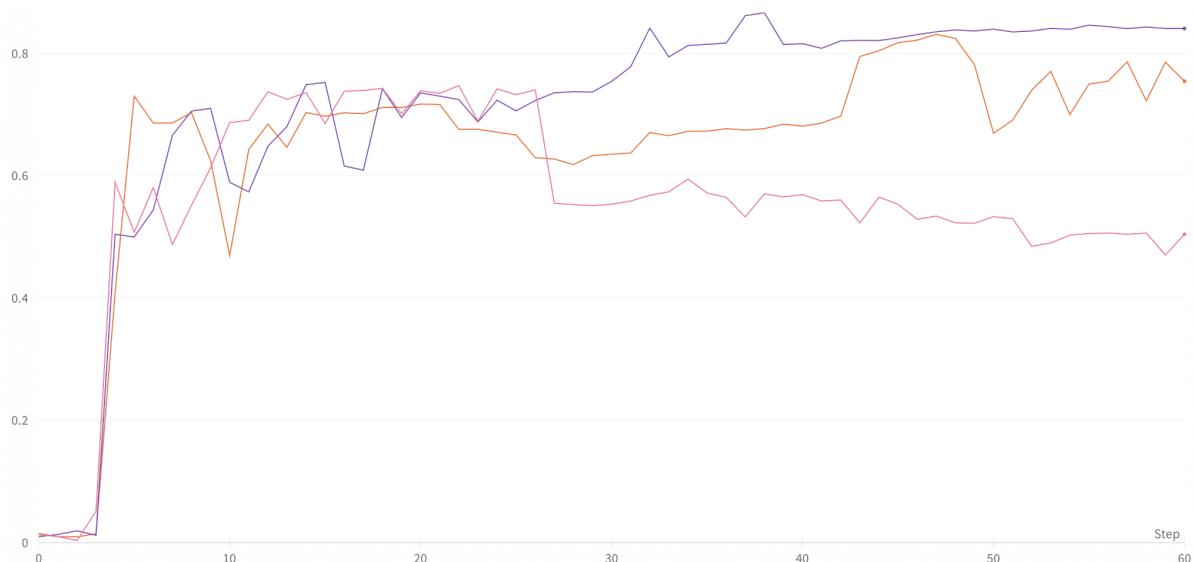
We hebben het YOLOv5 model getraind met deze nieuwe dataset. Vervolgens hebben we nog meer extra afbeeldingen gemaakt en gelabeld. Dit heeft een dataset opgeleverd waarbij iedere letter in ieder geval 24 keer vertegenwoordigd is. De Health check van Roboflow (Zie figuur 38) geeft nog steeds aan dat de dataset niet in balans is, echter is er bij de minst voorkomende letters hierin wel progressie geboekt.



Figuur 38 (Wandb.ai, z.d.)

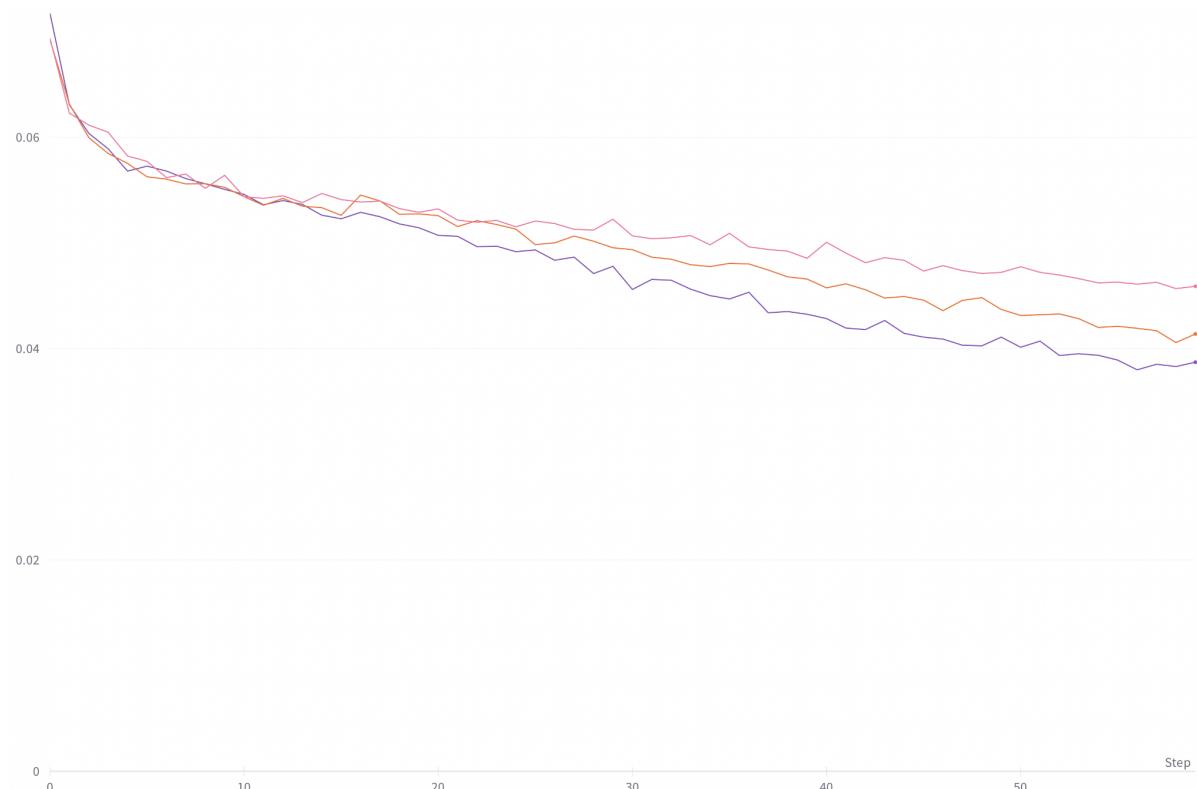
Ook met deze dataset hebben we het YOLOv5 model getraind. De precision grafiek, met op de x-as het aantal epochs en op de y-as de precision, is in figuur 39 te zien. De roze lijn is van het model getraind met de originele dataset, de paarse lijn met de iets uitgebreidere dataset en de oranje lijn met de nog iets uitgebreidere dataset.

Uit de grafiek blijkt dat wat betreft precision, de originele dataset het slechtst scoort. De uitgebreide datasets scoren hoger, waarbij de meest uitgebreide iets lager scoort dan de iets minder uitgebreide. Het is onduidelijk waarom de meest uitgebreide dataset minder hoog scoort dan de iets minder uitgebreide dataset, dit zou te maken kunnen met meetonzekerheid. Dat de originele dataset minder hoog scoort is wel verklaarbaar. Doordat de dataset meer gebalanceerd is, heeft het model minder bias. Het zou dan ook logisch zijn als de letters in de nieuwe afbeeldingen beter herkend worden. Als er meer letters correct worden voorspeld, heeft dit een hogere precision tot gevolg.



Figuur 39 (Wandb.ai, z.d.)

In de cls_loss grafiek zien we dat de originele dataset de hoogste loss heeft. Zoals beschreven in hoofdstuk 5.3 is het model minder goed in het bepalen van de class bij een hoge loss. Uit deze grafiek valt op te maken dat de uitgebreide datasets beter zijn in het bepalen van de class. Dit ligt in lijn met wat we zagen in de precision grafiek. Het toevoegen van nieuwe afbeeldingen lijkt dus een positief effect te hebben op de prestaties van het model.



Figuur 40 (Wandb.ai, z.d.)

5.6 Wat zijn de ethische gevolgen van het automatiseren van CAPTCHA's?

Bij het automatiseren van CAPTCHA's zijn er verschillende ethische gevolgen betrokken. In het kader van het geautomatiseerd oplossen van CAPTCHA's is er aan de ene kant het risico dat het middel misbruikt kan worden om CAPTCHA's te omzeilen. Op deze manier kan de veiligheid niet meer gewaarborgd worden op websites met dit veiligheids mechanisme. Aan de andere kant is het een oplossing om onleesbare en tijdrovende CAPTCHA's snel te kunnen oplossen.

5.6.1 Ethische theorieën

Bij een ethische stroming wordt er gekeken vanuit een bepaald perspectief. Er zijn verschillende ethische theorieën. Elke ethische theorie heeft specifieke kenmerken. Daarnaast is er bij elke ethische theorie een aantal sterktes en zwaktes. Bij ethiek is er niet één ethische theorie dat als de waarheid beschouwd kan worden (Welke ethische theorieën en stromingen zijn er?, 2018): "Er zijn veel verschillende ethische theorieën. Maar er is niet één theorie die 'waar' is."

De drie meest gangbare ethische stromingen zijn: Deugdethiek, Deontologie (Handelingsethiek) en Utilitarisme.

5.6.2 Deugdethiek

Vanuit de ethische stroming deugdethiek wordt er gezocht naar een gulden middenweg. Daarnaast wordt er gekeken naar het grotere geheel bij handelingen. Een bepaalde actie is daardoor niet een losstaande gebeurtenis. Zoals wordt bevestigd: (Agora theoriekamers, 2019): "Handelingen worden in deze ethiek nooit geïsoleerd bekeken, maar altijd beschouwd als een voortvloeisel uit het karakter van de actor. Mensen hebben in de loop van hun leven een karakter ontwikkeld, waarin bepaalde gewoonten en gebruiken zich hebben vastgezet, en deze zorgen dat zij op een specifiek moment tot een bepaald soort handelen zijn geneigd."

Een zwakte van deugdethiek is dat deze stroming niet gebonden is aan strikte regels. Bij deze ethische stroming wordt meer gekeken naar de context van acties. Zoals wordt beschreven is dit ook een zwakte van deze ethische stroming (Virtue Ethics: Strengths & Weaknesses – PHI220 Ethics, 2020): "On the other hand, some thinkers argue that virtue ethics provides vague and ambiguous advice. Because of its emphasis on the imprecise and highly contextual nature of ethics, virtue ethics is often criticized as insufficient as a guide to taking specific action".

Vanuit de deugdethiek stroming wordt er minder gekeken naar strikte regelgeving en andere mensen om het uiteindelijke doel te bereiken. In het kader van het geautomatiseerd oplossen van CAPTCHA's zal er een gulden middenweg gekozen moeten worden. Een voorbeeld van deze stroming is dat het automatisch oplossen van CAPTCHA's zelfstandig de juiste afwegingen maakt bij het oplossen. Dit houdt in dat er een mechanisme in kan zitten dat controleert voor welk doeleinde het middel gebruikt wordt. Dit mechanisme is aangeleerd gedrag en bevat geen strikte regelgeving waar het aan moet houden.

5.6.3 Deontologie

De ethische stroming deontologie ook wel Handelingsethiek is een stroming dat uitgaat van strikte regelgeving. Een handeling is vanuit deze ethiek onjuist op het moment dat het niet voldoet aan de geldende regelgeving. Dit wordt ook bevestigd (Welke ethische theorieën en stromingen zijn er?, 2018): "Daarbij doen de gevolgen van die handeling niet ter zake: ook al zijn die nog zo positief, een handeling is moreel onjuist als deze niet aan de geldende regel voldoet.". Daarnaast is bij deze ethische stroming dat de elke groep of individu gelijk is bij het oplossen van ethische dilemma's. Dit wordt ook beschreven (Welke ethische theorieën en stromingen zijn er?, 2018): "Een ander belangrijk beginsel uit de deontologie is dat je mensen nooit alleen als middel, maar ook altijd als doel in zichzelf moet zien. Zo mag je de ene mens niet opofferen om een ander te redden volgens Kant."

Een nadeel van deontologie is dat er niet altijd één universele lijst is met alle regels waar rekening gehouden moet worden. Daarbij komt dat het per onderwerp lastig te bepalen is wie welke regels op moet stellen. Dit wordt ook bevestigd: "Both also rely on absolute principles regarding duties and rights. But there's no definitive list recorded anywhere.". Een ander nadeel is dat er weinig ruimte is met betrekking van de context bij een handeling (Deontology: Strengths & Weaknesses – PHI220 Ethics, 2020): "In the case of duty-based ethics, people may object to the principle that people deciding on a course of action should ignore the circumstances in which they and other individuals find themselves. Duty ethics allows little room for context."

Vanuit de Deontologie stroming, in het kader van het automatisch oplossen van CAPTCHA'S, zou er strikt rekening gehouden moeten worden met alle bijkomende regelgeving. Bij het uitbrengen van dit middel kan het grote veiligheidsrisico's met zich mee brengen. Het uiteindelijke gevolg zou kunnen zijn dat het middel toegepast wordt om persoonsgegevens in te kunnen zien na het veelvuldig oplossen van wachtwoorden op websites. Websites bevatten het veiligheidsmechanisme van CAPTCHA om misbruik tegen te gaan, en te controleren of een persoon de gegevens invult. Door dit te omzeilen, is het veiligheidsmechanisme niet meer in werking. Doordat er persoonsgegevens ingezien kunnen worden zou dit in strijd zijn met de AVG wetgeving.

5.6.4 Utilitarisme

Bij de ethische stroming utilitarisme gaat het niet om de handeling of de intentie. Het gaat bij deze ethische stroming om de uiteindelijke consequenties. Als de consequentie goed is, is de handeling ook goed. Zoals wordt bevestigd: "Een utilitarist kijkt alleen naar de gevolgen van handelingen. Als het gevolg van de handeling goed is, is ook de handeling zelf goed. Het utilitarisme wordt daarom ook wel consequentialisme genoemd: alleen de consequentie is belangrijk.". Daarnaast wordt er gekeken bij de consequentie of er het voor een grote groep zoveel mogelijk 'geluk' oplevert. Dit wordt ook bevestigd (Utilitarisme, deontologie en deugdenleer: ethische systemen, 2019): "Wat de positieve gevolgen betreft, heeft hij het over "maximalisatie van geluk": de grootst mogelijke hoeveelheid geluk voor de grootst mogelijke groep mensen. Als er daarvoor een kleine minderheid mensen negatieve gevolgen moet ondervinden (moet lijden), dan is dat maar zo."

Een zwakte van deze ethische stroming heeft te maken met gerechtigheid. Omdat er vanuit deze ethische stroming zoveel mogelijk wordt gekeken naar de uiteindelijke consequentie voor een grote groep, kan dit zorgen voor onrecht voor een kleine groep of individu. Dit wordt ook bevestigd als zwakte van deze stroming (Utilitarianism: Strengths & Weaknesses – PHI220 Ethics, 2020): “It seems that a Utilitarian would be forced to accept that eliminating this minority would increase the happiness for the majority of people, and would therefore be a moral action. But this seems wrong, mainly because removing the minority from society would involve what many people take to be morally evil actions, which is another problem with Utilitarianism.”

In het kader van het geautomatiseerd oplossen van AI en het toepassen van de utilitarisme ethiek, wordt er voornamelijk gekeken naar de uiteindelijke consequentie van het uitbrengen van dit middel. Het gevolg van het uitbrengen van dit middel kan ervoor zorgen dat een grote groep kan profiteren van het snel ophalen of versturen van gegevens op websites. Het invullen van tijdrovende CAPTCHA's wordt voor deze grote groep opgelost. Maar aan de andere kant brengt het voor een grote groep ook grote veiligheidsrisico's mee. Door het middel in te zetten bij het oplossen van wachtwoorden kan dit zorgen voor een grote schade. Deze grote schade zou zoveel mogelijk beperkt moeten worden vanuit deze ethische stroming. Het middel zou daarom geoptimaliseerd moeten worden om de minste schade te kunnen oplopen bij het gebruik.

6 Discussie

6.1 Niet behaald

Uiteraard waren er een aantal punten die niet veel, of zelfs helemaal niet, aan bod zijn gekomen. Dit waren de drie belangrijkste punten die jammer genoeg niet uitgewerkt waren.

6.1.1 Gespiegelde letters

Het eerste punt dat was blijven liggen was waren de gespiegelde letters die niet lekker gingen. Bij het trainen van ons model waren er hyperparameters aanwezig die de afbeelding spiegelen. Hierdoor liepen de d/b & q/p niets zo lekker, want het model kon het verschil er niet tussen zien. Door dit probleem begon het model te gaan gokken tussen de twee letters en was de accuraatheid ook immens laag. We wouden dit probleem oplossen, echter wisten we niet hoe en zijn we hier later ook niet meer op teruggekomen.

6.1.2 Alternatieve modellen

Daarnaast merkten we dat we veel op YOLOv5, ons eerst werkende model, hebben gefocust waardoor we daarna niet meer modellen hebben getest. Op suggestie van Theo hebben wij toch diverse modellen getest: Faster R-CNN, Mask R-CNN, TensorFlow en DarkNet. Sommige leken geschikte resultaten te tonen, echter waren we al zo ver in het project dat we besloten hebben om toch niet van model te veranderen. Dit zou tijd besparen en ons meer ruimte geven om alle deadlines zorgvuldig te behalen.

6.1.3 Datasets

Ten slotte toen we merkte dat onze dataset te veel/te weinig aan bepaalde letters had dus kregen we de suggestie om een nieuwe dataset te zoeken. Omdat we al de huidige dataset in Roboflow gelabeld hadden, hadden wij eigenlijk weinig interesse om dit hele process weer te doen. Dus hebben wij dit laten liggen.

6.2 Wel behaald

De resultaten zouden anders geweest kunnen zijn als de bovenstaande punten anders waren aangepakt. Toch zijn we tevreden met het huidige resultaat.

6.2.1 Werkend model

Gelukkig zijn er wel resultaten die wel behaald zijn, onder andere we hebben een model werkend in YOLOv5. Dit model is een aantal uur getraind en kan nu dan ook de meeste letters accuraat lezen en voorspellen. Deze resultaten zijn met dank aan Data Augmentation wat we eerder hebben benoemd. Hierdoor is er meer dan genoeg data gegenereerd om de model mee te trainen. Door de grote hoeveelheid data worden de voorspellingen nauwkeuriger waardoor we uiteindelijk een waardevolle Proof of Concept hebben kunnen samenstellen.

6.2.2 Bevindingen

Gedurende het project zijn er bevindingen genoteerd door alle groepsleden. Deze bevindingen zijn genoteerd voor kleinere onderzoeken binnen het project. Een voorbeeld hiervan zijn de verschillende AI modellen die uitgeprobeerd zijn.

6.2.3 Onderzoeksverslag

Uiteindelijk hebben we alles van de bevindingen en Proof of Concept genoteerd in dit document, het onderzoeksverslag. Hierin hebben we onze hoofdvraag kunnen beantwoorden met behulp van onze deelvragen. En al deze deelvragen hebben we weer kunnen beantwoorden door vele bevindingen en onze Proof of Concept. Naast dat dit onderzoeksverslag de diverse vragen beantwoord geeft het ook inzicht in wat we deze 16 weken hebben bereikt en tot hoeverre we gegroeid zijn in het project zelf. Oorspronkelijk wisten we namelijk helemaal nog niets van AI en snapten wij stiekem ook nog niet wat voor resultaten we moesten opleveren. Echter begrijpen we nu al heel wat ervan en dit gehele process over hoe we hierop zijn gekomen hebben wij gezamenlijk bijgehouden in dit document.

6.3 Meegenomen kennis

Tijdens deze 16 weken is onze kennis erg veel gegroeid. Dit zijn de meest belangrijke punten die we willen meenemen naar toekomstige projecten rondom AI.

6.3.1 Voortzetten van trainen

In het begin van het project zijn we bij het trainen van een model bezig geweest om het aantal epochs te verhogen. Dit had een positieve invloed op de accuraatheid van het model. Google Colab heeft limitaties op de tijdsduur van het trainen van een model. Dit zorgde ervoor dat het model maximaal maar twee uur kon trainen. Het gebruik maken van een zwaarder model (met meer parameters) zorgde er al snel voor dat er niet verder getraind kon worden. Na een aantal weken in het project zijn we erachter gekomen dat het mogelijk is om het trainen voort te zetten van een eerdere poging. Door dit toe te passen was het mogelijk om een grote hoeveelheid epochs te gebruiken.

6.3.2 Data Augmentation

Bij het toepassen van Data Augmentation hebben we ervan geleerd dat het mogelijk is om meerdere afbeeldingen te genereren. Het stapsgewijs toepassen van uitgekozen data augmentation heeft ervoor gezorgd dat het model accurater is geworden.

6.3.3 Transfer Learning

We kwamen er al snel achter dat het gebruik van Transfer Learning erg handig is. Door dit toe te passen verloopt het trainen een stuk sneller. YOLOv5 biedt verschillende checkpoints aan die je kan gebruiken die al voor 300 epochs zijn getraind op de COCO dataset. Hierdoor is de trainingstijd veel minder en is de accuracy verbeterd.

6.3.4 Diverse Tools (Wandb/Roboflow/Google Colab)

6.3.4.1 Weights and Biases

Gedurende het project hebben we gebruik gemaakt van diverse tooling. Bij de tool WandB (Weights and Biases) hebben we geleerd om grafieken uit te lezen die gegenereerd zijn bij het trainen van een model. Door eerst de theorie te begrijpen van deze grafieken was het mogelijk om de juiste aanpassingen te maken aan het model.

6.3.4.2 Roboflow

Bij het gebruik van de tool Roboflow hebben we geleerd om data te labelen aan de hand van ‘bounding boxes’. Het voordeel van deze tool was dat het verdelen van taken en samenwerken goed mogelijk was. Door roboflow te gebruiken hebben we ook inzicht gekregen in de balans van de dataset. Bij sommige letters waren er onvoldoende labels aanwezig. Dit was terug te zien in het eindresultaat van het model. Deze weinig voorkomende letters werden minder vaak goed herkend dan meer voorkomende letters.

6.3.4.3 Google Colab

Bij het gebruik van Google Colab hebben we ervan geleerd om code op te splitsen in cellen. Door dit principe toe te passen is het mogelijk om code afzonderlijk uit te voeren. Dit zorgde er tevens voor dat het geheel overzichtelijk eruit zag. Daarnaast hebben we geleerd om een model eerst te trainen met Google Colab, en dit later als ‘pre-trained’ model te kunnen gebruiken. Tenslotte hebben we ervan geleerd om gericht te kiezen voor specifieke hardware bij het trainen van een model. Wij hebben namelijk gericht gekozen voor het gebruik van een GPU bij het trainen. Door middel hiervan is het mogelijk om berekeningen te paralleliseren. Bij een CPU wordt elke berekening na elkaar uitgevoerd. Het probleem van het gebruik van een CPU is dat het trainen van een model langer duurt.

6.4 Onverwachte veranderingen

Natuurlijk waren er een aantal punten tijdens het project die plotseling naar boven kwamen. Dit waren onze onverwachte veranderingen waar we snel op aangepast hebben.

6.4.1 Labelen van data

Toen we begonnen met het labelen van data deden we dit door de CAPTCHA's in zijn geheel op te lossen. We kwamen er na het labelen achter dat dit niet zou gaan werken. De resultaten van een eerste test die we deden waren erg slecht en Theo adviseerde ons om de letters individueel te labelen. We hebben Roboflow hiervoor gebruikt en we hebben het labelen verdeeld over het team.

6.4.2 Van Tensorflow naar PyTorch

In de eerste versie gebruikte we een model wat gebruik maakte van Tensorflow. We kwamen toen YOLOv5 tegen, dat is een port van YOLOv5 van Tensorflow naar PyTorch. PyTorch heeft namelijk betere performance en is de laatste tijd steeds populairder aan het worden (Connor, 2021). De switch was makkelijk te maken omdat er niet veel was gewijzigd.

6.4.3 Bevindingen

Bevindingen staat zowel in dit rijtje als bij ‘Wel behaald’, omdat we halverwege het project pas realiseerde hoe belangrijk bevindingen waren. En dat we tot toen nog helemaal niets noteerde over onze geteste modellen en/of belangrijke nieuwe kennis. We noteerde kort onze notulen van verschillende gesprekken, maar een apart mapje op onze Google Drive voor bevindingen was ook belangrijk. Hierin schreven we eigenlijk alles wat ons was opgevallen tijdens verschillende processen zoals het testen van een model. Om niet alles het ergens genoteerd te hebben, maar ook om kennis te delen met de rest van groep. Zoals er al verteld was, deze bevindingen zijn uiteindelijk ook meegenomen naar het onderzoeksverslag.

7 Conclusie

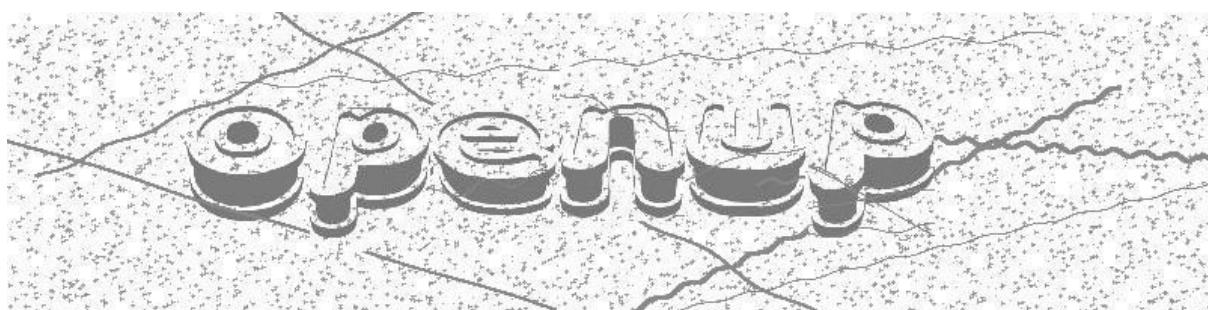
We hebben aangetoond dat het mogelijk is om CAPTCHA's automatisch op te lossen met een Deep Learning model, maar dit werkt niet perfect. We zagen de beste resultaten met het YOLOv5 model. Om een goede accuraatheid te krijgen was het wel nodig om een dataset van 400+ afbeeldingen te labelen en gebruik te maken van Data Augmentation. Ook varieert de trainingstijd tussen de 1 en 5 uur. Zelfs na de optimalisaties zagen we wel dat het model niet perfect is. Het is vooral lastig om balans te krijgen in de dataset waardoor je een bias hebt voor bepaalde letters.

We hebben ons ook gerealiseerd dat er ethische gevolgen zitten aan dit model omdat CAPTCHA's zijn gemaakt om websites te beschermen van automatische software. We denken dat dit aantoon dat CAPTCHA's tegenwoordig niet meer veilig genoeg zijn en eigenlijk alleen de gebruiksvriendelijkheid verminderen. Dit baseren wij op het feit dat wij al snel zonder enige voorkennis over AI een fatsoenlijke Proof of Concept hebben kunnen opstellen. Een alternatief als Two-Factor Authentication in plaats van CAPTCHA's zou bijvoorbeeld een mooie oplossing kunnen zijn, omdat je hier afhankelijk bent van externe authenticatie zoals via e-mail of telefoon. Hierbij kan een AI (nog) niet bij en dus vinden wij 2FA geschikter dan CAPTCHA's.

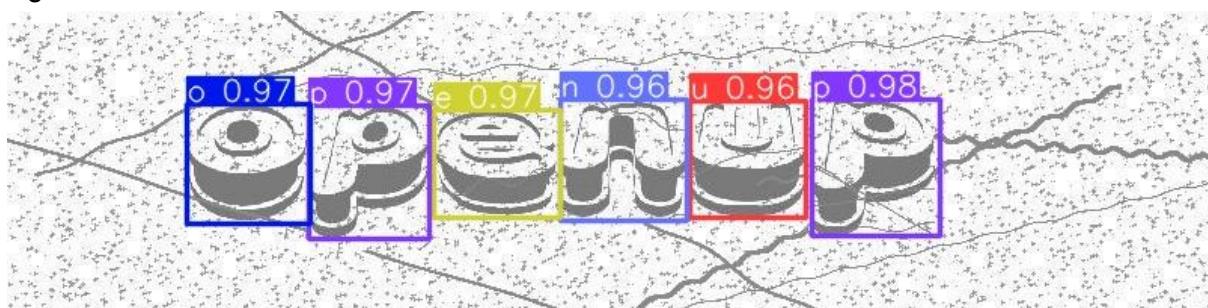
7.1 Proof of Concept

Uiteindelijk hebben wij gekozen voor YOLOv5 omdat dit model ons de beste accuraatheid en snelheid opleverde. Na urenlang het model trainen kan ons Proof of Concept diverse CAPTCHA's met een goede zekerheid oplossen. Daarnaast hebben wij dit model nog getest met unieke plaatjes, die niet uit onze gegeven dataset komen. Onze Proof of Concept kan [hier](#) gevonden worden.

7.1.1 Eigengemaakte woord



Figuur 41

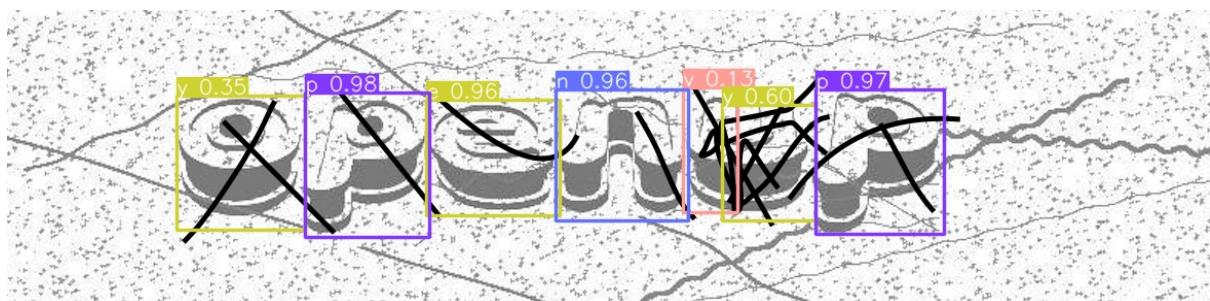


Figuur 42

7.1.2 Eigengemaakte woord met strepen erdoorheen.

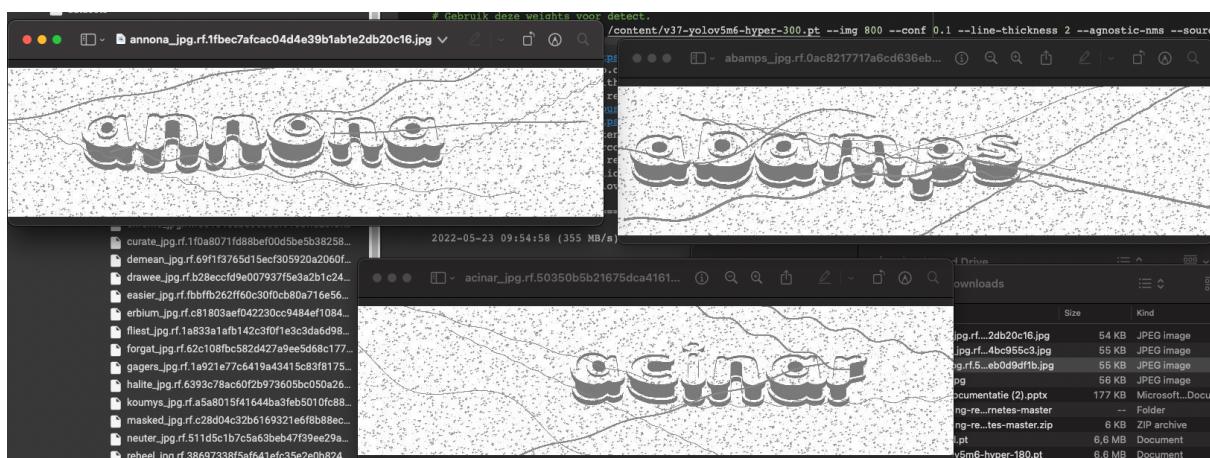


Figuur 43

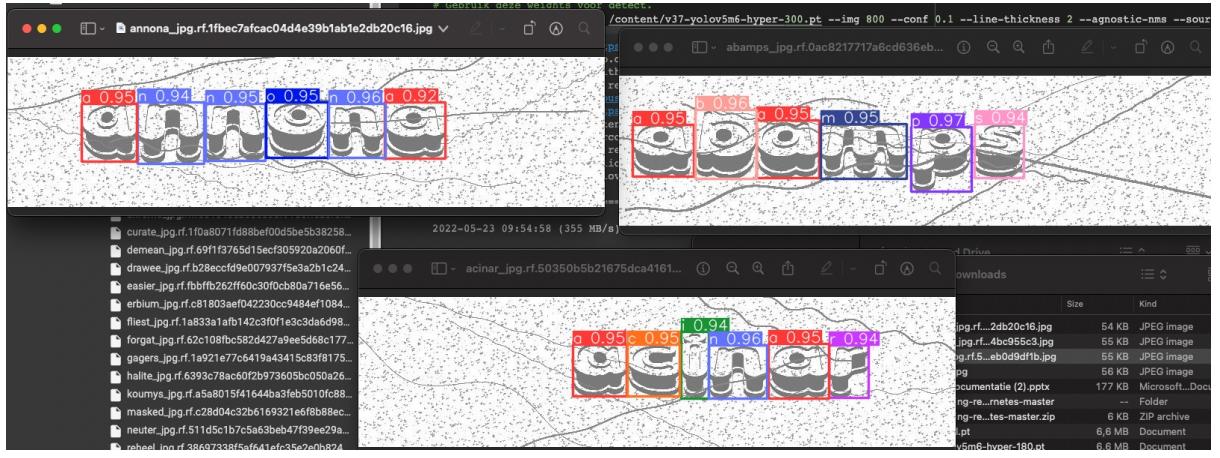


Figuur 44

7.1.3 Meerdere CAPTCHA's in 1 plaatje.



Figuur 45



Figuur 46

Alle Google Colabs

Hier zijn al onze Google Colabs te vinden die wij hebben gebruikt of getest tijdens dit project.

Models

Dit zijn de Colab bestanden waarin we diverse modellen hebben getest, om ze met elkaar te vergelijken.

- **Captcha OCR:**
https://colab.research.google.com/drive/15cjym5XMVFSC-WymH_KrS9UXCxaRYieP
- **Mask R-CNN:**
<https://colab.research.google.com/drive/1Kwg-f3heeyUPXO-wdc62jtzA-CETif0U>
- **Faster R-CNN**
<https://colab.research.google.com/drive/1n5oO6tFm7OAV3JKWczlVL9eHh9DSLgB8>
- **EfficientDet:**
<https://colab.research.google.com/drive/1NU7eo2Bkcr6CLye3JAqvl4VxE12qrgSW>
- **EfficientDet v2:**
https://colab.research.google.com/drive/1SzH0j2ZJtvYzA7JNH_Z4EDAqEo9hawSc
- **YOLOv5:**
<https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb>

Trainen

Deze Colab bestanden zijn gebruikt om te experimenteren met het trainen van een model. Om deze uit te voeren moet je vaak lang trainen, soms uren. Maar het kan wel interessant zijn om te zien hoe de modellen zijn getraind.

- **YOLOv5 Onderzoek #1: Alex**
https://colab.research.google.com/drive/1qdOP1b3ZhmULcFarnRF_hCxQiDBI484o
- **YOLOv5 Onderzoek #2: Auke**
<https://colab.research.google.com/drive/1WRgQ9T1sofSHrCVbfJq1-kDLKzV9KLC>
- **YOLOv5 Onderzoek #3: Steven**
<https://colab.research.google.com/drive/1V1o4E68yFJOvwacIfHZxurrlx3OVi4XA>
- **YOLOv5 Onderzoek #4: Mike**
<https://colab.research.google.com/drive/16npReA0IT6oMXGK1YrTdx0OzsqYvZ8G7>

Detecteren

Deze Colab bestanden hebben wij gemaakt om de voortgang in een toegankelijke manier te laten zien. Je kan deze notebooks in hun geheel uitvoeren. We gebruiken hiervoor getrainde

- **Data Augmentation:**
<https://colab.research.google.com/drive/1Fi0Zr2zo4KwBF069jD6PNsAJyXFU-xb7>
- **Hyper Parameters + Betere dataset:**
https://colab.research.google.com/drive/1mBAF4HnDa7vF-d_13OO15da6_W7UhjH2
- **YOLOv5 modellen:**
<https://colab.research.google.com/drive/1qEeW7MRwObVaOOuEmAqaZH5D6bRbZmEX>
- **Eindresultaat (Proof of Concept):**
https://colab.research.google.com/drive/1JILNy03KpM_E9oyB9TpQ-CFjR9vNj93i

Literatuurlijst

Scott, E. (z.d.). CAPTCHAs Have an 8% Failure Rate, and 29% if Case Sensitive – Articles

–. Baymard Institute. Geraadpleegd op 22 maart 2022, van

<https://baymard.com/blog/captchas-in-checkout>

Boag, P. (2017, 1 augustus). My definitive guide to why CAPTCHA sucks. Paul Boag - User Experience Advice. Geraadpleegd op 10 mei 2022, van

<https://boagworld.com/usability/my-definitive-guide-to-why-captcha-sucks/>

Ensie. (2022, 10 mei). Ethiek | betekenis & definitie. Geraadpleegd op 10 mei 2022, van

<https://www.ensie.nl/betekenis/ethiek>

Hendrickson, J. (2022, 25 maart). The Great Cyberwar Has Just Begun: You Need to Protect Yourself. ReviewGeek. Geraadpleegd op 10 mei 2022, van

<https://www.reviewgeek.com/113090/the-great-cyberwar-has-just-begun-you-need-to-protect-yourself/>

Trendskout. (2021, december 2). De verhouding tussen AI, ML en DL schematisch weergegeven. [Foto]. Artificial Intelligence, Machine Learning en Deep Learning: wat is het verschil? <https://trendskout.com/general/ai-vs-machine-learning-vs-deep-learning/>

IBM Cloud Education. (2021, 16 september). Artificial Intelligence (AI). IBM. Geraadpleegd op 10 mei 2022, van <https://www.ibm.com/cloud/learn/what-is-artificial-intelligence>

Oracle. (z.d.). Wat is AI? Meer informatie over kunstmatige intelligentie. Geraadpleegd op 10 mei 2022, van <https://www.oracle.com/nl/artificial-intelligence/what-is-ai/>

Brownlee, J. (2019, 6 augustus). Train Neural Networks With Noise to Reduce Overfitting. Machine Learning Mastery. Geraadpleegd op 10 mei 2022, van

<https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/>

Brownlee, J. (2019, 17 juni). What is the Difference Between a Parameter and a Hyperparameter? Machine Learning Mastery. Geraadpleegd op 31 mei 2022, van
<https://machinelearningmastery.com/difference-between-a-parameter-and-a-hyperparameter/>

Ng, A. (2019, 16 oktober). Deep Learning Curve [Foto]. What is Deep Learning? <https://machinelearningmastery.com/what-is-deep-learning/>

Grossfeld, B. (2020, 2 september). Deep Learning versus zelflerende systemen: een eenvoudige manier om het verschil te begrijpen. Zendesk NL. Geraadpleegd op 10 mei 2022, van <https://www.zendesk.nl/blog/machine-learning-and-deep-learning/#georedirect>

Education, I. C. (2022, 30 maart). Deep Learning. Ibm. Geraadpleegd op 10 mei 2022, van <https://www.ibm.com/cloud/learn/deep-learning>

Oracle. (z.d.-b). Wat is machine learning? Geraadpleegd op 10 mei 2022, van <https://www.oracle.com/nl/data-science/machine-learning/what-is-machine-learning/>

Rooijakers, S. (2022, 12 januari). Wat is Deep Learning? SDIM. Geraadpleegd op 10 mei 2022, van <https://www.sdim.nl/helpcentrum/begrippenlijst/deep-learning/>

Saxena, S. (2021, 12 maart). Image Augmentation Techniques | Image Augmentation for Deep Learning. Analytics Vidhya. Geraadpleegd op 10 mei 2022, van <https://www.analyticsvidhya.com/blog/2021/03/image-augmentation-techniques-for-training-deep-learning-models/>

Nkomo, B. (2022, 14 februari). Convolutional Neural Networks — Part 1: Edge Detection. Medium. Geraadpleegd op 10 mei 2022, van <https://medium.com/swlh/convolutional-neural-networks-22764af1c42a>

#003 CNN More On Edge Detection. (2018, 1 november). [Foto]. #003 CNN More On Edge Detection. <https://datahacker.rs/edge-detection-extended/>

Dwivedi, P. (2021, 15 december). YOLOv5 compared to Faster RCNN. Who wins? - Towards Data Science. Towards Data Science. Geraadpleegd op 10 mei 2022, van <https://towardsdatascience.com/yolov5-compared-to-faster-rcnn-who-wins-a771cd6c9fb4>

A Guide to RNN: Understanding Recurrent Neural Networks and LSTM Networks. (2021, 29 juni). [Foto]. Builtin. <https://builtin.com/data-science/recurrent-neural-networks-and-lstm>

Maindola, G. (2021, 26 augustus). A Brief History of YOLO Object Detection Models From YOLOv1 to YOLOv5. MLK - Machine Learning Knowledge. Geraadpleegd op 10 mei 2022, van <https://machinelearningknowledge.ai/a-brief-history-of-yolo-object-detection-models/>

DeepAI. (2020, 25 juni). Stride (Machine Learning). Geraadpleegd op 10 mei 2022, van <https://deeppai.org/machine-learning-glossary-and-terms/stride>

Telus. (2022, 17 januari). What's the difference between CNN and RNN? Geraadpleegd op 10 mei 2022, van <https://www.telusinternational.com/articles/difference-between-cnn-and-rnn>

Inc, W. (2022, 16 februari). Why GPUs for Machine Learning? A Complete Explanation. WEKA. Geraadpleegd op 10 mei 2022, van <https://www.weka.io/blog/gpus-for-machine-learning/>

Jocher, G. (2022, 10 april). Train Custom Data [Foto]. <https://github.com/ultralytics> <https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data>

Sagar, R. (2022, 9 februari). A Beginner's Guide To TPUs. Analytics India Magazine. Geraadpleegd op 10 mei 2022, van <https://analyticsindiamag.com/tpu-beginners-guide-google/>

Google. (z.d.). Google Colab. Geraadpleegd op 10 mei 2022, van
<https://research.google.com/colaboratory/faq.html>

Holmes, J. (2022, 26 januari). Comparison of Basic Deep Learning Cloud Platforms - Geek Culture. Geek Culture. Geraadpleegd op 10 mei 2022, van
<https://medium.com/geekculture/comparison-of-basic-deep-learning-cloud-platforms-337657edb710>

DeVries Et Al. (2017). Papers with Code - Cutout Explained. Paperswithcode. Geraadpleegd op 10 mei 2022, van <https://paperswithcode.com/method/cutout>

Ghoneim, S. (2021, 9 december). Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on? Medium. Geraadpleegd op 10 mei 2022, van
<https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124>

Agora theoriekamers. (2019). Theoriekamers. Geraadpleegd op 16 mei 2022, van
<http://theoriekamers.nl/pages/nl-achtergrond-deugden-ethiek.aspx>

Welke ethische theorieën en stromingen zijn er? (2018, 14 februari). KNMG. Geraadpleegd op 16 mei 2022, van
<https://www.knmg.nl/advies-richtlijnen/ethische-toolkit/verdiepen/welke-ethische-theorieen-en-stromingen-zijn-er-1.htm>

Virtue Ethics: Strengths & Weaknesses – PHI220 Ethics. (2020, 30 oktober). Pressbooks. Geraadpleegd op 17 mei 2022, van
<https://viva.pressbooks.pub/phi220ethics/chapter/virtue-ethics-pros-cons/>

Normen en waarden: Ethiek! (2021, 29 november). Rijksuniversiteit Groningen. Geraadpleegd op 17 mei 2022, van
https://www.rug.nl/society-business/scholierenacademie/scholieren/pws-hulp/verkennen-en-opzetten/alle-schoolvakken-en-hulp/normen-en-waarden_-ethiek_

Microsoft Docs. (2021, 18 november). Confidence score - QnA Maker - Azure Cognitive Services. Geraadpleegd op 17 mei 2022, van
<https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/concepts/confidence-score>

Rajput, M. (2020, 2 juli). YOLO V5 — Explained and Demystified. Towards AI. Geraadpleegd op 17 mei 2022, van
<https://towardsai.net/p/computer-vision/yolo-v5%E2%80%8A-%E2%80%8Aexplained-and-demystified>

Xu, R. (2021, februari). The network architecture of Yolov5 [Foto]. Researchgate.
[https://www.researchgate.net/figure/The-network-architecture-of-Yolov5-It-consists-of-three-parts-1-Backbone-CSPDarknet_fig1_349299852#:~:text=It%20consists%20of%20three%20parts%3A%20\(1\)%20Backbone%3A%20CSPDarknet.score%2C%20location%2C%20size](https://www.researchgate.net/figure/The-network-architecture-of-Yolov5-It-consists-of-three-parts-1-Backbone-CSPDarknet_fig1_349299852#:~:text=It%20consists%20of%20three%20parts%3A%20(1)%20Backbone%3A%20CSPDarknet.score%2C%20location%2C%20size)

Utilitarianism: Strengths & Weaknesses – PHI220 Ethics. (2020, 30 oktober). Pressbooks. Geraadpleegd op 17 mei 2022, van
<https://viva.pressbooks.pub/phi220ethics/chapter/utilitarianism-pros-and-cons/>

Roboflow: Give your software the power to see objects in images and video. (z.d.). Roboflow. Geraadpleegd op 17 mei 2022, van <https://roboflow.com/>

Grandperrin, J. (2021, 27 december). How to use confidence scores in machine learning models. Medium. Geraadpleegd op 17 mei 2022, van <https://towardsdatascience.com/how-to-use-confidence-scores-in-machine-learning-models-abe9773306fa>

Google. (z.d.-a). Classification: Precision and Recall [Illustratie]. Classification: Precision and Recall. <https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall>

Utilitarisme, deontologie en deugdenleer: ethische systemen. (2019). InfoNu. Geraadpleegd op 17 mei 2022, van <https://mens-en-samenleving.infonu.nl/filosofie/173783-utilitarisme-deontologie-en-deugdenleer-ethische-systemen.html>

Deontology: Strengths & Weaknesses – PHI220 Ethics. (2020, 30 oktober). Pressbooks. Geraadpleegd op 17 mei 2022, van <https://viva.pressbooks.pub/phi220ethics/chapter/deontology-pros-cons/>

Brems, M. (2020, 11 december). 5 Strategies for Handling Unbalanced Classes. Roboflow Blog. Geraadpleegd op 17 mei 2022, van <https://blog.roboflow.com/handling-unbalanced-classes/>

JavaFX Shearing - javatpoint. (z.d.). Www.Javatpoint.Com. Geraadpleegd op 17 mei 2022, van <https://www.javatpoint.com/javafx-shearing>

Dataset Health Check - Roboflow. (z.d.). Roboflow Docs. Geraadpleegd op 17 mei 2022, van <https://docs.roboflow.com/dataset-health-check>

Entropie: wat is dat precies? - Mr. Chadd Academy. (z.d.). Mrchadd. Geraadpleegd op 17 mei 2022, van <https://www.mrchadd.nl/academy/vakken/natuur-leven-en-techniek/entropie-wat-is-dat-precies>

Roboflow. (z.d.). [Illustratie]. Roboflow. <https://app.roboflow.com/>

Weights & Biases - Documentation. (z.d.). Weights & Biases. Geraadpleegd op 17 mei 2022, van <https://docs.wandb.ai>

YOLOv5 - Documentation. (z.d.). Weights & Biases. Geraadpleegd op 17 mei 2022, van <https://docs.wandb.ai/guides/integrations/yolov5>

Man vs machine: comparing artificial and biological neural networks. (2017, 21 september). [Illustratie]. Sophos. <https://news.sophos.com/en-us/2017/09/21/man-vs-machine-comparing-artificial-and-biological-neural-networks/>

Solawetz, J. (2020, 20 augustus). Why and How to Implement Random Rotate Data Augmentation. Roboflow Blog. Geraadpleegd op 18 mei 2022, van <https://blog.roboflow.com/why-and-how-to-implement-random-rotate-data-augmentation/#:%E7E:text=One%20common%20data%20augmentation%20technique,to%20encompass%20the%20resulting%20object>

An experiment with YOLOv3-4-5 and EfficientDet for infrastructure asset management – Result! Data. (2020, 7 juli). ResultData. Geraadpleegd op 23 mei 2022, van <https://www.resultdata.ai/an-experiment-with-yolov3-4-5-and-efficientdet-for-infrastructure-asset-management/>

GitHub - ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite. (z.d.). GitHub. Geraadpleegd op 23 mei 2022, van <https://github.com/ultralytics/yolov5>

Dwivedi, P. (z.d.). YOLOv5 vs. Faster RCNN, who wins and who loses? (Reproduced) - Fear Cat. Fear Cat. Geraadpleegd op 23 mei 2022, van <https://blog.fearcat.in/a?ID=01750-0c41eabc-1adb-4c88-bb46-a457396019fe>

Connor, R. (2021, 14 december). PyTorch vs TensorFlow in 2022. AssemblyAI Blog. Geraadpleegd op 31 mei 2022, van <https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2022/>

Hyperparameter Evolution - YOLOv5 Documentation. (z.d.). YOLOv5 Documentation. Geraadpleegd op 31 mei 2022, van <https://docs.ultralytics.com/tutorials/hyperparameter-evolution/>