Urban Point Cloud Classification

Classificatie van objecten in het straatbeeld aan de hand van point clouds met behulp van machine learning

Auteur	Job Jonkers	Bedrijf:	Appnormal
Studentnummer:	421580	Bedrijfsbegeleider:	Roy van der Veen
Opleiding:	HBO-ICT SE	Onderwijsinstelling:	Saxion Enschede
Datum:	13-06-2021	Afstudeerbegeleider:	René van den Nieuwenhoff

Voorwoord

In opdracht van Appnormal wordt er onderzoek gedaan naar de mogelijkheden van machine learning technieken met point clouds van stedelijke omgevingen. Het doel is om een systeem te creëren dat volautomatisch statistieken van objecten in het straatbeeld kan berekenen. Dit onderzoek wordt uitgevoerd in het kader van het afstuderen van de HBO-ICT Software Engineering opleiding aan het Saxion in Enschede. De afstudeerperiode loopt van februari 2021 tot en met juni 2021. Graag wil ik Roy van der Veen van Appnormal en René van den Nieuwenhoff van het Saxion bedanken voor de uitstekende begeleiding tijdens het afstudeertraject.

Samenvatting

In opdracht van Appnormal is tijdens dit afstudeertraject onderzoek gedaan naar het verkrijgen van informatie uit het straatbeeld met behulp van point clouds en machine learning. Een point cloud is een verzameling van vaak honderdduizenden punten in een 3D ruimte. Deze point clouds kunnen een omgeving of object belichamen. Door point clouds te verwerken met een machine learning model kan er allerlei informatie verkregen worden zonder menselijke interactie. In de toekomst kan Appnormal samen met bezorgdienst “Tring Tring” 3D omgevingsdata van het straatbeeld gaan verzamelen met behulp van scanners die point clouds opleveren. De hoofdvraag van het afstudeerproject is “Hoe kan Machine Learning ingezet worden om voor gemeentes relevante objecten en statistieken uit het straatbeeld te extraheren uit point clouds?” Deze objecten en statistieken kunnen afgestemd worden op de eindgebruiker. Een voorbeeld van objecten en statistieken uit een straatbeeld is de hoeveelheid prullenbakken die in een point cloud gevonden worden.

Na een analysefase waarin praktijkgericht onderzoek is gedaan naar machine learning technieken en een ontwerpfase waarin de opzet van het systeem is gemaakt, is het systeem ontwikkeld in de realisatiefase van het afstudeertraject. Het systeem bestaat uit twee aaneenschakelingen van machine learning technieken, deze aaneenschakelingen worden pipelines genoemd. Er zijn twee pipelines ontwikkeld: de train-pipeline en de inference-pipeline. Met de train-pipeline kan op basis van point clouds met bijgevoegde informatie over de inhoud een machine learning model getraind worden. Met de inference-pipeline kan met behulp van een reeds getraind model informatie uit een point cloud geëxtraheerd worden. Aan deze pipelines zijn toevoegingen gedaan om een zo goed mogelijk resultaat te krijgen. Zo worden punten die niet bijdragen aan de informatie van de point cloud weg gefilterd en worden er bewerkingen uitgevoerd op de training data. Hiermee wordt het systeem robuuster gemaakt voor point clouds van wisselende kwaliteit. Het gerealiseerde systeem kan op basis van de beschikbare data met een zekerheid van rond de 80 procent de punten en objecten in een point cloud classificeren.

De conclusie is daarom dat het mogelijk is om een systeem te bouwen waarmee point clouds van het straatbeeld verwerkt kunnen worden tot nuttige informatie met behulp van machine learning. De resultaten hiervan zijn echter afhankelijk van de diversiteit en kwaliteit van de beschikbare point clouds. Het resultaat kan verbeterd worden door het gebruik van een meer diverse en grotere dataset. Daarnaast kunnen de voorafgaand aan het trainen uitgevoerde acties geoptimaliseerd worden. Het opgeleverde systeem vormt een goede basis voor doorontwikkeling en het toepassen van deze verbeteringen.

Begrippenlijst

Begrip	Definitie/verklaring
Accuracy	De precisie waarmee het model punten kan classificeren ten opzichte van de vooraf gelabelde punten die als waarheid worden gezien.
Augmentation	Het veranderen van de punten in de point cloud om zo een iets andere versie van de point cloud te krijgen, vaak met als doel om de hoeveelheid beschikbare data te vergroten.
Class	De naam voor een type object dat in de point cloud voorkomt, bijvoorbeeld een prullenbak of verkeersbord
Class instance	Als in een point cloud een class vaker voorkomt, bijvoorbeeld één point cloud met drie prullenbakken, dan worden deze losse prullenbakken 'instances' van deze class genoemd.
Classificeren	Verzamelnaam voor methodes waarmee van punten in een point cloud bepaald kan worden wat deze belichamen of waar deze onderdeel van zijn.
Convolutional Neural Network (CNN)	Type ML model dat geïnspireerd en gemodelleerd is naar de menselijke visie. Wordt veelal gebruikt voor problemen met visuele toepassingen zoals afbeeldingen.
Downsampling	Het al dan niet willekeurig uitdunnen van een point cloud door punten te verwijderen.
Gelabelde data	Point clouds waarvan duidelijke is welke punten samen een object vormen. Het ML model kan deze informatie gebruiken om objecten te herkennen
Ground truth	De gelabelde punten van de point cloud die gezien worden als de juiste punten en dus als de 'waarheid'.
Inference	Het gebruiken van een getraind machine learning model voor gevolgtrekking. Bij inference classificeert objecten in een point cloud.
Labelen	Het markeren van punten in een point cloud die bij een bepaald object of bepaalde 'class' horen.
Loss	Waarde die tijdens het trainen aangeeft hoe slecht de schattingen van het model zijn. Hoe hoger de loss, hoe slechter de inschattingen.
Machine learning (ML)	Verzamelnaam voor technieken waarmee computers kunnen leren om specifieke problemen op te lossen
Model	Machine Learning model. Netwerk van machine learning technieken dat getraind kan worden en conclusies kan trekken uit abstracte data.
Neural network	Type ML model dat geïnspireerd en gemodelleerd is naar de menselijke hersenen en zenuwsysteem.
Outlier removal	Het verwijderen van punten die niet in de buurt van andere punten liggen.
Point cloud	Verzameling punten in een 3-dimensionale ruimte die samen een geheel vormen. Point clouds kunnen naast ruimtelijke informatie ook kleurinformatie bevatten.
PointCNN	Een point cloud machine learning model dat gebruik maakt van een Convolutional Neural Network om point clouds te classificeren

RGB	Systeem om een kleur uit te drukken met drie waardes. Een RGB point cloud is een point cloud met kleurinformatie.
Trainen	Het "leren" van het machine learning model om bepaalde patronen te ontdekken op basis van de ingevoerde gelabelde data

Inhoudsopgave

1. Inleiding

Machine learning is een innovatieve technologie waarmee ogenschijnlijk abstracte data kan omgezet worden tot bruikbare informatie. Vooral op het gebied van machine learning met point clouds zijn nog weinig concrete toepassingen. Het automatisch verwerken van point clouds met machine learning biedt de mogelijkheid om allerlei verschillende soorten informatie te verkrijgen zonder menselijke interactie. Een goed getraind machine learning model kan met hoge precisie omgevingsinformatie ontfutselen uit ruwe point clouds. Hiermee kunnen onder andere menselijke inspecties van omgevingen geautomatiseerd worden. Hoewel deze techniek nog redelijk in de kinderschoenen staat zijn de mogelijkheden interessant en divers. De focus van dit afstudeertraject lag op het onderzoeken en ontwikkelen van een systeem dat point clouds omzet tot interessante informatie uit het straatbeeld.

1.1 Appnormal

Appnormal is een ICT-bedrijf dat zich specialiseert in het ontwikkelen van slimme app oplossingen, het optimaliseren van bedrijfskundige processen en het denken vanuit gebruikersperspectief. Hierbij maken ze gebruik van de laatste technieken om apps te bouwen die gebruikerservaring en design combineren. Appnormal bestaat 11 jaar en is in die tijd uitgegroeid tot een bedrijf met 11 werknemers.

1.2 Leeswijzer

Hoofdstuk 2 biedt een overzicht van het probleem met hierbij de start- en gewenste situatie, het doel, de afbakening en de onderzoeksvragen die naar aanleiding hiervan zijn opgesteld. In hoofdstuk 3 worden vervolgens aan de hand van de probleemstelling de requirements en stakeholders verder toegelicht. Hierna gaat hoofdstuk 4 in op de procesmethodiek en de structuur van de verschillende fases van het afstudeertraject. Hoofstuk 5 staat in het teken van de resultaten van de analysefase. In hoofdstuk 6 staat de ontwerpfase en de resultaten hiervan centraal. Hoofdstuk 7 presenteert de resultaten van de realisatiefase. Op basis van de voorgaande hoofdstukken wordt vervolgens in hoofdstuk 8 een conclusie getrokken. Tot slot bevat hoofdstuk 9 een lijst met aanbevelingen voor de opdrachtgever.

2. Probleemstelling

2.1 Startsituatie

In de toekomst kan Appnormal samen met de bezorgdienst TringTring 3D omgevingsdata van het straatbeeld gaan verzamelen met behulp van LIDAR-scanners. Een LIDAR-scanner kan een 3D representatie maken van een omgeving door de afstand tot objecten in het gezichtsveld te meten. Deze gegevens worden opgeslagen als punten die vervolgens samen een point cloud kunnen vormen. Door het gebruik van Machine Learning kunnen we gegevens over het straatbeeld uit deze point clouds halen.

De bezorgers van TringTring fietsen veel door de binnenstad van onder andere Amsterdam. Door scanners op de fietsen van deze bezorgers te monteren kan er point cloud data van deze omgeving worden verzameld. Door een machine learning model te bouwen en te trainen om objecten in het straatbeeld te herkennen, kunnen we gegevens van deze objecten uit scans halen. Deze gegevens kunnen interessant zijn voor de gemeente van de stad waar de scans gemaakt zijn. Bij het starten van de afstudeerperiode was er point cloud data beschikbaar van straten en losse objecten in het straatbeeld.

2.2 Gewenste situatie

In de gewenste situatie zal het mogelijk zijn om een scan in te laden in een systeem dat vervolgens de nuttige informatie van de objecten in de scan toont in een interface. Om tot deze nuttige informatie te komen moet de point cloud verwerkt worden door een aaneenschakeling van machine learning technieken. Ook moet het mogelijk zijn om op basis van scans het machine learning model te trainen. Het getrainde model kan vervolgens worden gebruikt om de point cloud te verwerken tot nuttige informatie. Bij nuttige informatie over deze objecten kan bijvoorbeeld gedacht worden aan afmetingen, locatie en hoe vaak ze voorkomen in het straatbeeld.

2.3 Doel

Het doel van de opdracht is het bouwen van een systeem dat data kan opleveren die bruikbaar is voor gemeentes. Gedurende het afstudeertraject zal gekeken worden welke objecten in het straatbeeld voor Appnormal interessant en haalbaar zijn om te verwerken. Dit zal grotendeels afhankelijk zijn van de resultaten van het systeem en de wensen van Appnormal over de informatie die het eindproduct van dit project in eerste instantie moet opleveren. Bij objecten uit het straatbeeld kan gedacht worden aan statische objecten zoals prullenbakken of verkeersborden, of tijdsgebonden data zoals de gemiddelde parkeerdrukte en hoeveelheid afval. Gemeentes kunnen deze gegevens gebruiken om hun binnenstad beter in kaart te krijgen. De gebruiker zal door middel van het labelen zelf de mogelijkheid hebben om te bepalen welke informatie in de scans als belangrijk gezien wordt.

2.4 Afbakening

Het is belangrijk om de opdracht af te bakenen zodat het duidelijk is wat wel en niet binnen de scope van de opdracht valt. Het systeem hoeft niet in realtime de data te kunnen verwerken. Dit betekent dat we niet tijdens, maar na afloop van het scannen de point clouds pas zullen verwerken.

2.5 Onderzoeksvragen

Vanuit de probleemanalyse is de volgende hoofdvraag gedefinieerd.

Hoofdvraag: *Hoe kan Machine Learning ingezet worden om voor gemeentes relevante objecten en statistieken uit het straatbeeld te extraheren uit point clouds?*

De onderstaande deelvragen stippen de verschillende onderwerpen aan die komen kijken bij het beantwoorden van de hoofdvraag. Om de hoofdvraag te kunnen beantwoorden zijn de volgende deelvragen gedefinieerd:

Deelvraag 1: *Hoe kunnen de ruwe point clouds voorbereid worden zodat ze geschikt zijn voor het trainen van een machine learning model?*

Voordat een point cloud gebruikt kan worden voor het trainen van een model zullen er een paar stappen gezet moeten worden. Denk hierbij aan de indeling en het bestandsformaat van de point cloud en het labelen van de informatie waarmee getraind moet worden.

Deelvraag 2: *Hoe kan een machine learning model geïmplementeerd worden in een pipeline met als doel het automatiseren van de verwerking van point clouds?*

Om de resultaten van deelvraag 1 toe te kunnen passen moet er een machine learning model worden geïmplementeerd. Dit model kan na het antwoorden van deelvraag 2 met de beschikbare data van deelvraag 1 trainen en inference uitvoeren.

Deelvraag 3: *Hoe kunnen er met behulp van de geëxtraheerde objecten statistieken over het straatbeeld gevormd worden?*

Het doel van het systeem is het verwerken van de point clouds tot informatie over het straatbeeld. Alleen de output van inference van een machine learning model is hierbij niet genoeg. Statistieken kunnen aan de hand van de resultaten van het machine learning model gegenereerd worden.

3. Casus analyse

In dit hoofdstuk wordt aan de hand van de probleemstelling in hoofdstuk 2 de casus verder toegelicht. Hierbij wordt er gekeken naar de requirements en stakeholders van de casus. Op basis van de stakeholders van het project kunnen de requirements opgesteld worden.

3.1 Stakeholders

De stakeholders zijn personen of organisaties die een belang hebben bij de uitvoering van deze afstudeeropdracht. Voor deze afstudeeropdracht is het aantal stakeholders vrij beperkt. In principe zijn er drie stakeholders: Appnormal, Saxion en de afstudeerder zelf. Appnormal is een stakeholder omdat deze de opdrachtgever is van het project. Appnormal wordt voornamelijk gerepresenteerd door de bedrijfsbegeleider Roy van der Veen. Vanuit de bedrijfsbegeleider worden de wensen van Appnormal toegelicht en wordt de voortgang van het project geobserveerd en besproken met de afstudeerder. Voor Appnormal is vooral het praktische eindproduct van dit afstudeertraject van belang, de bijbehorende documentatie is voor deze stakeholder hier ondergeschikt aan. Appnormal is een belangrijke stakeholder en beoefent actief invloed uit op de koers van het afstudeertraject. Na afronding van het afstudeertraject kan Appnormal de ontwikkeling van het product voortzetten en op de markt brengen. Omdat het project bedoeld is om een indicatie te geven van de mogelijkheden van de technologie is er nog geen contact gelegd met mogelijke eindgebruikers zoals gemeentes. Indirect behartigt Appnormal dus ook de belangen van deze eindgebruikers. Daarnaast is Hogeschool Saxion ook een stakeholder in dit project, deze voert vooral een controlerende rol uit. De begeleider vanuit het Saxion, René van den Nieuwenhoff, is hierbij de vertegenwoordiging van deze stakeholder. Voor Saxion als stakeholder is vooral het afstudeerdossier van belang. Hieronder vallen de op te leveren documenten zoals de scriptie en andere niet-schriftelijke beroepsproducten. De laatste stakeholder is de afstudeerder zelf, deze tracht de best mogelijke uitkomst van het afstudeerproject te realiseren door met de andere stakeholders samen te werken.

3.2 Requirements

Op basis van gesprekken met de stakeholders zijn de eerste ruwe requirements opgesteld. Vervolgens zijn deze in de ontwerpfase verder uitgewerkt met de opgedane kennis van de analysefase. Hierdoor was het mogelijk om vrij concrete en technisch gedetailleerde requirements op te stellen. Zo zijn er ook requirements opgesteld die niet direct vanuit de stakeholders zijn opgelegd maar op basis van het onderzoek wel belangrijk kunnen zijn voor het project. De requirements zijn opgesplitst tussen de Train-pipeline en Inference-pipeline. Over deze pipelines is meer informatie te vinden in Hoofdstuk 6.

3.2.1 Train-pipeline requirements

	Requirement
1	De pipeline heeft gelabelde point clouds als input met een specifiek bestandstype
2	Input point clouds worden omgezet naar een bruikbaar formaat voor het trainen van het model
3	Outlier removal kan uitgevoerd worden op de input data
4	De pipeline maakt gebruik van scaling augmentations
5	De pipeline maakt gebruik van rotation augmentations
6	De pipeline maakt gebruik van noise augmentations
7	De pipeline maakt gebruik van downsampling augmentations
8	Gebruiker heeft de keuze om wel of niet augmentations en/of outlier removal te gebruiken
9	De augmentations en outlier removal zijn geoptimaliseerd voor de beschikbare dataset

10	De input data moet na eventuele bewerkingen zoals augmentations gesplitst worden in train en test/validatie batches
11	Het machine learning model kan getraind worden met de beschikbare data
12	Het machine learning model toont de resultaten van het trainen met daarbij de behaalde accuracy en loss
13	De statistieken van het trainen van het model worden getoond in Tensorboard
14	Het getrainde machine learning model wordt geëxporteerd
15	Het is mogelijk om met behulp van een settings-bestand de (hyper)parameters van het model en de instellingen van de pipeline te bepalen
16	De code in de notebook wordt ondersteund door middel van documentatie
17	De Train-pipeline is ook beschikbaar als python bestand om uitgevoerd te worden in de command line

3.2.2 Inference-pipeline requirements

	Requirement
1	De pipeline heeft ongelabelde point clouds van een bepaald bestandstype als input
2	De pipeline kan gelabelde point clouds van een bepaald bestandstype als input gebruiken
3	De pipeline verwerkt de input point cloud tot een bruikbaar formaat voor inference
4	De pipeline kan gebruik maken van outlier removal op de input point cloud
5	De gebruiker heeft de keuze om wel of niet outlier removal op de input point cloud uit te voeren
6	De pipeline kan een door de Train-pipeline getraind model importeren voor gebruik bij de inference
7	Het is mogelijk om te kiezen welk model geïmporteerd wordt
8	Met het model kan inference op de input data uitgevoerd worden
9	De resultaten van inference moeten als point cloud bestand geëxporteerd kunnen worden
10	De resultaten van inference kunnen geëvalueerd worden als de input point cloud een gelabelde point cloud is
11	Aan de hand van het inference resultaat moeten er statistieken over de point cloud berekend worden
12	Het is mogelijk om met behulp van een settings-bestand de (hyper)parameters van het model en de instellingen van de pipeline te bepalen
13	De code in de notebook wordt ondersteund door middel van documentatie
14	De Inference-pipeline is ook beschikbaar als python bestand om uitgevoerd te worden in de command line

4. Procesmethodiek

Om tijdens het afstudeertraject structuur aan te brengen en overzicht te behouden is er gekozen om te werken aan de hand van bepaalde methodes en processen. In dit hoofdstuk wordt toegelicht hoe de procesmethodiek is toegepast.

4.1 Procesbeschrijving

De afstudeerperiode kan opgedeeld worden in twee delen, de uitvoering en de afronding. Bij elk deel horen verschillende deliverables, deze zijn in de Gantt chart onder het kopje Planning terug te vinden in de Gantt chart. De uitvoering vindt plaats vanaf het begin van de afstudeerperiode tot ongeveer vier weken voor het einde. De laatste vier weken van het afstuderen vallen onder de afronding.

4.1.1 Uitvoering

Omdat de uitvoering een lange periode betreft, is deze opgedeeld in drie aparte projectfases: Analyse, Ontwerp en Realisatie. Aan elk van deze periodes is een hoofdstuk gericht waarin de resultaten van deze periode worden besproken. Daarnaast is er voor de Realisatiefase een Realisatieverslag (zie Bijlage 1) opgeleverd waarin deze fase in meer detail wordt toegelicht. Dit verslag is als bijlage toegevoegd. In elk van de drie fases is er naast de werkzaamheden die onderdeel uitmaakten van die fase ook gewerkt aan de documentatie.

Analyse

In deze fase lag de focus op het analyseren van de beschikbare materialen in de startsituatie en het onderzoeken van onderwerpen en technieken die tijdens de ontwerp- en realisatiefase van pas zouden komen. Dit onderzoek was zowel op het gebied van literatuur als het gebied van empirisch onderzoek in de vorm van experimenteren met technieken die aan bod kwamen tijdens deze fase. Hierbij heeft praktijkgericht onderzoek voorrang gehad over literatuuronderzoek.

Ontwerp

De ontwerpfase was een vrij korte fase van slechts 2 weken. In deze fase is met de opgedane kennis van de analysefase een ontwerp uitwerken voor de op te leveren producten. Naast het opzetten van deze ontwerpen is deze fase ook benut om het Kanban bord met alle taken op te zetten en in te delen.

Realisatie

De realisatiefase was de langste en meest belangrijke fase van de uitvoering. Tijdens deze fase zijn de deliverables op softwaregebied gerealiseerd. Bij de ontwikkeling hiervan is er gebruik gemaakt van het Kanban systeem dat tijdens de ontwerpfase is opgezet.

4.1.2 Onderzoeksmethodiek

Om te kunnen bepalen of de onderzochte informatie van belang is voor het afstudeerproject is het verstandig om vast te houden aan bepaalde onderzoeksmethoden. Omdat de afstudeeropdracht voornamelijk praktijkgericht is, zijn de gebruikte onderzoeksmethoden hier ook op afgestemd. De focus van de gebruikte methoden ligt dan ook minder op literatuuronderzoek en meer op het vergaren van praktische informatie door middel van praktijkgericht en empirisch onderzoek. Onderzoeksmethoden die hier goed op aansluiten zijn onder andere Exploratief onderzoek, deskresearch, experimenteel onderzoek en toegepast onderzoek. Bij Exploratief onderzoek ligt de nadruk vooral op het uitvoeren van verkennend onderzoek, voorafgaand hiervan is er nog geen goed beeld van de resultaten. Dit type onderzoek wordt gebruikt om ideeën op te doen en richting te

geven aan het vervolgonderzoek. Voor dit vervolgonderzoek kan gebruik worden gemaakt van deskresearch en experimenteel onderzoek. Voor deskresearch worden secundaire gegevens gebruikt om te bepalen welke informatie of technieken geschikt kunnen zijn voor de realisatiefase. Een goed voorbeeld hiervan is het vergelijken van verschillende machine learning modellen op basis van de prestaties op verschillende datasets. Hier sluit de experimentele onderzoeksmethode goed op aan. Bij deze methode wordt het effect van een bepaalde aanpassing of techniek toegepast en bestudeerd. Deze methode wordt onder andere gebruikt bij de outlier removal en augmentation technieken in de analysefase die in Hoofdstuk 5 worden beschreven. Afhankelijk van de uitkomst van deze onderzoeken die op basis van deze methoden worden gedaan kan bepaald worden of de onderzochte onderwerpen waardevol kunnen zijn in de realisatiefase.

4.2 Procesmanagement

Zoals reeds benoemd is de realisatiefase ingericht aan de hand van Kanban taken die tijdens de ontwerpfase zijn opgesteld. De keuze is gevallen op de Kanban methode omdat deze, in tegenstelling tot Scrum, veel vrijheid geeft over de indeling van de taken. Voor veel van de taken was het lastig om een goede aanduiding te geven over de tijd die nodig was om de taak af te ronden. Kanban sloot hier goed op aan door flexibel te zijn in het tijdsbestek waarin taken afgerond moeten worden. Ook het gebrek aan sprints zoals we die van Scrum kennen is daarom een voordeel van Kanban.

De Kanban taken bestaan uit een titel, een omschrijving en afbakening van de werkzaamheden en een checklist met punten die afgerond moeten zijn voordat de story afgerond kan worden. Daarnaast is er ook een Definition-of-Done opgezet die in principe voor alle taken geldt. Dit zijn de eisen waaraan het resultaat van elke taak aan moet voldoen voordat deze als ‘Done’ gemarkeerd kan worden. De Definition of Done is als volgt:

- Alle taken op de checklist van de story zijn afgerond
- De code van de story is handmatig getest
- De code in de Notebook wordt ondersteund door tekst en documentatie
- In het geval van problemen geeft de code een duidelijke foutmelding terug

De Definition-of-Done is niet heel uitgebreid, dit is vooral omdat de opdracht door een individu is uitgevoerd in plaats van een team. De individuele taken worden niet per stuk beoordeeld of ‘gereviewd’, wel wordt de opgeleverde code bij het wekelijkse voortgangsgesprek besproken.

5. Onderzoeksresultaten

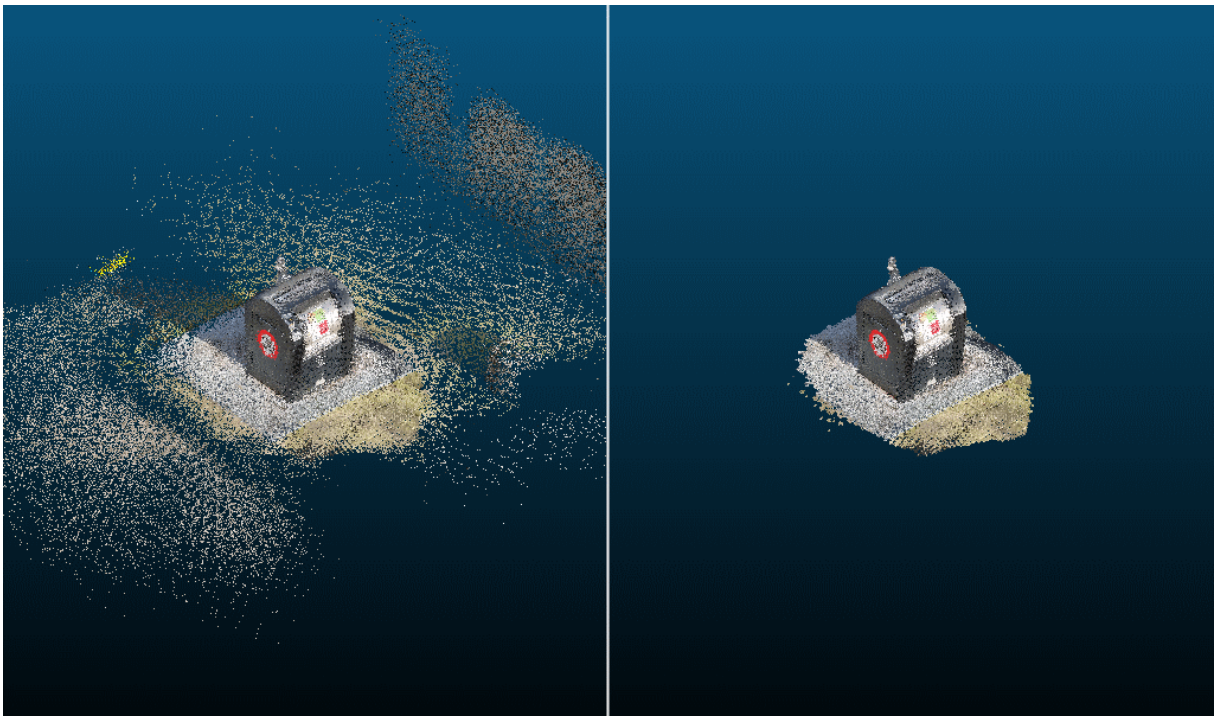
In dit hoofdstuk worden de onderzoeksresultaten van de analysefase toegelicht en de deelvragen beantwoord. Tijdens de analysefase is er praktijkgericht onderzoek gedaan naar verscheidene technieken die tijdens de realisatiefase gebruikt konden worden. Het antwoord op de hoofdvraag is te vinden in hoofdstuk 9.

5.1 Analysefase

In de 4 weken durende analysefase is er onderzoek gedaan naar machine learning technieken die in een later stadium van het afstuderen gebruikt konden worden. Dit onderzoek was praktijkgericht en bestond uit het maken van kleine 'proof of concepts' van de onderzochte technieken.

5.1.1 Optimaliseren van point clouds uit LIDAR-scans

De point clouds die voor het afstudeertraject beschikbaar zijn, zijn van redelijke kwaliteit. Toch is er winst te behalen door de point clouds te optimaliseren. Voor het trainen van het model is het namelijk wenselijk dat het aandeel van punten dat gewenste informatie bevat zo hoog mogelijk is. Er kan gesteld worden dat punten in de point cloud die niet in de buurt van andere punten liggen, minder informatie bevatten dan punten die wél naburige punten hebben. De punten zonder naburige punten noemen we 'outliers' en zijn niet wenselijk om mee te trainen. Om deze outliers te verwijderen kunnen we gebruik maken van 'outlier removal'.



Figuur 1 - Point cloud van een afvalcontainer voor (links) en na agressieve outlier removal

Met deze techniek kunnen we outliers wegfilteren op basis van twee parameters: een standaarddeviatie en de hoeveelheid nabije punten om te controleren. Er worden stuk voor stuk willekeurige punten uit de point cloud geselecteerd, voor elk geselecteerde punt worden de hoeveelheid nabije punten gecontroleerd. Als de afstand van deze punten groter is dan de opgegeven standaarddeviatie zullen ze verwijderd worden. In Figuur 1 hierboven is een vrij agressieve vorm van outlier removal te zien die tijdens de onderzoeksfase is gerealiseerd. Ervan

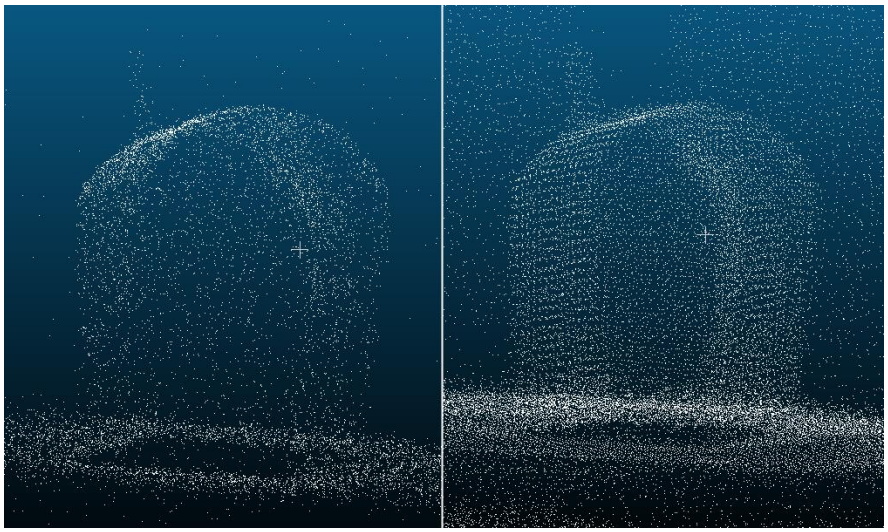
uitgaande dat alle beschikbare point clouds van vergelijkbare kwaliteit zijn dan is het mogelijk om bij benadering geschikte parameters te vinden voor de outlier removal. In de realisatiefase is er naar aanleiding van dit onderzoek gekeken naar de agressiviteit van de outlier removal en de invloed hiervan op de prestaties van het trainen van het model. Hierop wordt in hoofdstuk 7 en in het realisatieverslag dieper ingegaan.

5.1.2 Downsampling en nabootsing van point clouds van mindere kwaliteit

De beschikbare point clouds zijn, hoewel niet uitzonderlijk, van prima kwaliteit. Dit komt deels omdat de point clouds van individuele objecten zijn gemaakt in tegenstelling tot de hele straten die in een later stadium gebruikt zullen worden in het systeem. Een ander verschil is dat de beschikbare point clouds 360 graden rondom de objecten gemaakt zijn. De point clouds die vanaf een fiets en dus vanuit één richting gemaakt worden zullen niet alle zijden van een object bevatten. Ook zijn de huidige beschikbare point clouds gemaakt in goede weersomstandigheden. Point clouds die in regenachtige weersomstandigheden zijn gemaakt zullen hoogstwaarschijnlijk van lagere kwaliteit zijn. Het is dus aannemelijk dat het systeem na het afstudeertraject met point clouds van wisselende kwaliteit zal gaan werken. Deze kwestie gaat hand in hand met downsampling, een techniek waarbij het aantal punten in de point cloud verminderd wordt. Door het trainen met zowel de originele point clouds als gedownsamplede versies hiervan kan de robuustheid van het model verhoogd worden. Downsampling kan ongeacht de kwaliteit van de gebruikte data worden gebruikt om de prestaties van het model te verhogen op die data.

Het idee van downsampling is dus het uitdunnen van de point cloud, bij voorkeur met behoud van de structuur van de point cloud. Een eenvoudige manier van downsampling is willekeurige downsampling, waarbij willekeurig een aantal punten uit de point cloud wordt verwijderd. Een vergelijkbare variant is de nth row downsampling, hierbij wordt elke zoveelste rij in de array van punten van de point cloud verwijderd. Beide methodes geven geen controle over de punten die verwijderd worden, slechts over de hoeveelheid punten. Hierdoor is het niet gegarandeerd dat de point cloud zijn structuur behoudt, vooral bij agressieve downsampling. De resultaten van deze twee methodes zijn zo goed als vergelijkbaar.

Een duurzamere methode is ‘voxelgrid downsampling’, hierbij wordt de point cloud in een denkbeeldig 3D raster verdeeld en wordt er in elk vak van het raster één centraal punt gegenereerd. Door de grootte van de vakken van het raster te veranderen kan er invloed uitgeoefend worden op de mate van downsampling. Deze methode behoudt de structuur van de point cloud en garandeert een accurate representatie van de originele point cloud na het downsamplen.



Figuur 2- Random downsampling(links) en voxelgrid downsampling

In Figuur 2 is een point cloud te zien die gedownsampled is met zowel random downsampling als de voxelgrid methode. Het verschil tussen de twee methodes is hier duidelijk te zien. Waar de random downsampling geen duidelijk patroon heeft is dit bij het resultaat van de voxelgrid methode wel duidelijk te zien. Vooral bij agressieve downsampling is het behouden van de structuur zoals de voxelgrid methode dat doet erg belangrijk. Tijdens het onderzoek is gebleken dat voxelgrid downsampling met behoud van kleurinformatie niet mogelijk is. Dit komt omdat de originele punten van de point cloud vervangen worden door punten die gegenereerd worden in het raster. Het berekenen van een kleur voor het gegenereerde punt aan de hand van de kleuren van de originele punten in het raster is ook onderzocht. Het resultaat hiervan gaf echter in de meeste gevallen geen accurate representatie van de originele kleuren en is daarom niet toereikend. In hoofdstuk 7 zal er toegelicht worden hoe downsampling is toegepast in de pipelines.

5.1.3 Genereren van meer data door augmentation

Een machine learning model gedijt goed bij het trainen met voldoende representatieve data. De hoeveelheid beschikbare data voor dit afstudeertraject is, hoewel toereikend, wel enigszins beperkt. Om de hoeveelheid data te vergroten kan er gebruik worden gemaakt van ‘augmentations’. Augmentations zijn simpelweg kleine veranderingen aan de punten van een point cloud waardoor een iets andere point cloud ontstaat. Door het toepassen van meerdere augmentations op de beschikbare data kan de hoeveelheid data vermenigvuldigd worden. Een kanttekening hierbij is wel dat te veel augmentations van dezelfde point cloud kunnen gaan leiden tot ‘overfitting’. Hierbij is een model te veel getraind met een bepaalde soort data wat mindere prestaties bij het gebruik van andere data tot gevolg heeft.

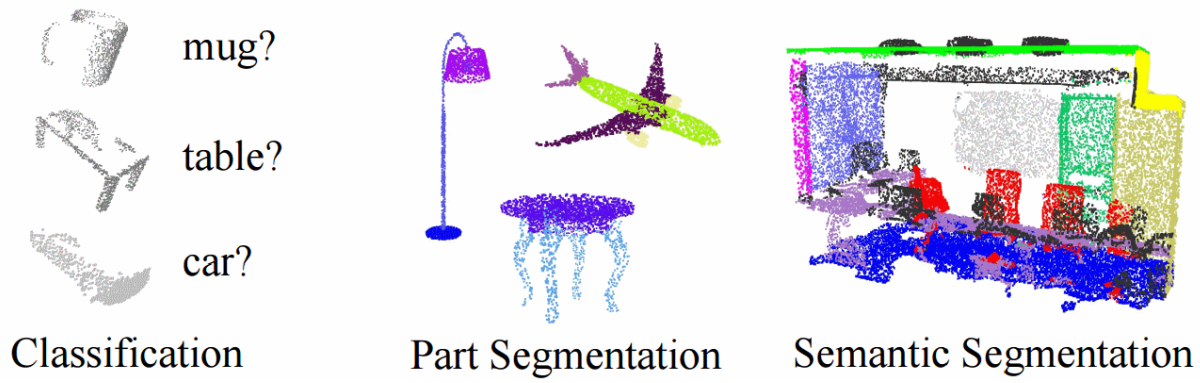
Tijdens de onderzoeksfase is er onderzoek gedaan naar verschillende soorten augmentations. Hierbij kan onderscheid gemaakt worden tussen augmentations waarbij de structuur van de point cloud onveranderd blijft en waarbij deze wél aangepast wordt. Twee voorbeelden van augmentations met behoud van structuur zijn ‘scaling’ en ‘rotations’. Bij scalen worden de locaties van de punten in de point cloud geïnterpoleerd of geëxtrapoleerd met als resultaat een point cloud die kleiner of groter dan het origineel is. Bij rotations wordt de originele point cloud enigszins gedraaid langs een of meerdere assen waardoor het resultaat een iets gedraaide variant van het origineel is. In het realisatieverslag wordt dieper ingegaan op de techniek achter deze varianten van augmentation

Een van de augmentations waarbij de structuur van de point cloud wel veranderd is 'noisen'. Bij deze techniek worden de locaties van de punten enigszins veranderd. Dit wordt gedaan met een voor elk punt willekeurig gekozen afwijking binnen een ingestelde marge. Door de marge aan te passen is het mogelijk om de mate van noise in het eindresultaat aan te passen.

Het is mogelijk om meerdere augmentation methodes te combineren om zo een grote hoeveelheid extra training data te creëren. Afhankelijk van de agressiviteit van de toegepaste augmentations wordt de originele point cloud meer of minder vertroebelt. De meest effectieve manieren en sterkte van augmentations verschilt per dataset en machine learning model. In hoofdstuk 7 wordt toegelicht hoe de augmentation technieken toegepast worden in de Train-pipeline en hoe tot een geschikte combinatie en agressiviteit van augmentations is gekomen.

5.1.4 Verschillende point cloud ML models en transfer learning

Een interessant onderzoeksonderwerp tijdens de analysefase was het vergelijken van verschillende machine learning modellen en de mogelijkheid tot transfer learning. Om te kunnen beginnen met het vergelijken van verschillende machine learning modellen die met point clouds werken, moet vastgesteld worden wat het machine learning model precies moet kunnen doen. Om specifieker te zijn: welk soort inference moet het model op de point clouds kunnen uitvoeren.



*Figuur 3 - Drie verschillende methoden van inference op point clouds. (*Classification, Part segmentation en semantic segmentation*, 2017)*

In Figuur 3 hierboven zijn drie verschillende point cloud inference methodes te zien: classification, part segmentation en semantic segmentation. Deze drie methodes hebben allemaal als doel het bepalen of punten ergens bij horen of niet, maar hebben elk andere uitkomsten. Classification is simpelweg het bepalen welk object het point cloud belichaamt. Hierbij wordt ervan uitgegaan dat alle punten in de point cloud bij dat object horen. Part segmentation is het onderscheiden van verschillende onderdelen van een point cloud, zonder dat hierbij wordt bepaald wat de onderdelen moeten voorstellen. Part segmentation gaat dan ook puur over het maken van de splitsing tussen de onderdelen. Daarnaast is er ook nog Semantic segmentation, deze techniek tracht alle objecten in een point cloud de ontdekken en deze tevens te classificeren als een bepaald object. Bij inference met semantic segmentation is het model in staat om alle punten in een point cloud onder een bepaalde class te scharen.

Voor het doeleinde van deze afstudeeropdracht is Semantic segmentation de meest geschikte methode van inference. Dit komt omdat in de point clouds die gebruikt worden meerdere objecten bevatten die interessant zullen zijn voor de uitkomst van het systeem. Daarnaast is het belangrijk om te weten welk soort objecten er in de point cloud te vinden zijn.

Het opzetten en testen van een machine learning model met de beschikbare data bleek een groot deel van het afstudeerproject te zijn. Het was dan ook simpelweg niet mogelijk om tijdens de onderzoeksfase meerdere machine learning modellen op te zetten en te testen. De keuze voor het gebruikte model is dan ook gemaakt op basis van de prestaties van de verschillende modellen op gerenommeerde datasets zoals S3DIS (Armeni et al., 2016) en ShapeNet (ShapeNET, 2021). Er is gekeken naar modellen als PointNet (Qi et al., z.d.), PointNet++ (Qi, R., Yi, et al., z.d.), RandLA-Net (Hu, z.d.), PointCNN (Li et al., z.d.) en RSNets (Huang et al., z.d.). Bij het vergelijken van de modellen is er naast de prestaties op bestaande datasets ook gekeken naar de kwaliteit van de documentatie en de technieken waarop het machine learning model gebaseerd is. Ook moest het model open source zijn zodat het geïmplementeerd kan worden in het afstudeerproject. Uiteindelijk is de keuze gevallen op PointCNN, dit model dat gebaseerd is op PointNet en PointNet++ heeft verreweg de beste documentatie en heeft een logische opbouw van code. Daarnaast heeft dit model samen met RandLA-Net, dat een prima alternatief van PointCNN is, over het algemeen de beste prestaties op veelgebruikte point cloud datasets. De implementatie van PointCNN in de pipelines wordt beschreven in het realisatie hoofdstuk.

5.1.4.1 Trainen van een model

Het trainen van een machine learning model is een wiskundig en ingewikkeld proces. Toch is dit proces in de basis ogenschijnlijk simpel. Een neuraal netwerk bestaat uit meerdere ‘neurons’ die aan elkaar verbonden zijn. Een netwerk heeft een beginpunt en een eindpunt met daarin de neurons die elk meerdere neurons als input en output hebben. Elke neuron voert een bewerking uit op dat data die het binnenkrijgt en stuurt het vervolgens door naar de volgende neurons in het netwerk. Elke neuron heeft ‘weights’ en ‘biases’, deze waardes hebben invloed op hoe de data in het neuron wordt verwerkt. Afhankelijk van hoe goed de inschatting van een neuron is krijgt deze neuron een ‘loss’ waarde aangewezen. Hoe lager deze waarde is, hoe beter de inschatting is geweest. Het doel van het neurale netwerk is het minimaliseren van de loss door aanpassingen te maken aan de weights en biases van elke neuron. Door meerdere malen door de training data te itereren worden de weights en biases steeds beter afgesteld op de gebruikte data. Hier zit echter ook een gevaar. Indien er te lang getraind wordt op een bepaalde dataset of soort data zullen deze waardes te veel worden afgestemd op die data. Dit wordt ‘overfitting’ genoemd en heeft negatieve prestaties op onbekende datasets tot gevolg. Overfitting kan worden opgespoord aan de hand van het vergelijken van de train- en validatieprestaties tijdens het trainen. Wanneer de accuracy van het trainen blijft stijgen terwijl de accuracy van de validatie die gelijktijdig plaatsvindt dat niet meer doet, is de kans groot dat er overfitting plaats vindt.

5.1.4.2 Transfer learning

Transfer learning is een bekende techniek binnen machine learning, waarbij een model deels getraind wordt op bestaande datasets om zo een goede basis te leggen voor verdere training. Het model kan vervolgens verder getraind worden met de specifieke dataset van de gebruiker. Tijdens het onderzoek bleek dat dit principe op het gebied van 3D point cloud machine learning vrijwel nergens wordt toegepast. Ook viel op dat de weinige onderzoeken naar transfer learning voor point

cloud machine learning vaak afgeschermd waren. Hierdoor is de keuze gemaakt om geen verder onderzoek naar transfer learning te doen.

5.2 Onderzoeksvragen

5.2.1 Deelvraag 1: Hoe kunnen de ruwe point clouds voorbereid worden zodat ze geschikt zijn voor het trainen van een machine learning model?

Voor het verwerken van een point cloud zodat deze gebruikt kan worden met een machine learning model zijn meerdere stappen nodig. Met een ruwe point cloud wordt er in dit geval vanuit gegaan dat deze bestaat uit punten met elk X, Y en Z coordinaten en kleurinformatie in de vorm van R, G en B waardes. De eerste stap in de voorbereiding is het 'labelen' van de point cloud, hierbij wordt van elk punt aangegeven van welke 'class' deze onderdeel is. Door het labelen kan bijvoorbeeld onderscheid gemaakt worden tussen welke punten bij een prullenbak horen en welke bij een paaltje horen. Het labelen kan gedaan worden met behulp van een tool die de gebruiker de mogelijkheid geeft om de punten van een bepaalde class te selecteren en die vervolgens het desbetreffende label toe te wijzen. Door dit labelen krijgt de point cloud een extra kolom met waardes, namelijk het nummer dat overeenkomt met de class van het label. In het verlengde hiervan kan er ook op 'instances' gelabeld worden. Hierbij wordt er rekening gehouden met meerdere objecten van dezelfde class in een point cloud. Bij het labelen van class instances krijgt de point cloud nog een extra kolom, waarbij elke instance van de class een eigen waarde krijgt toegekend. Voor het labelen is de Semantic Segmentation Editor (Mandrioli, z.d.) gebruikt, een open-source tool die het labelen van classes en instances in point clouds mogelijk maakt. Afhankelijk van de gebruikte tool kan het bestandsformaat van de input point cloud verschillen. Het labelen is een handmatige stap in dit proces. Het labelen is een vrij arbeidsintensieve taak. Dit hoeft echter alleen gedaan te worden wanneer het model getraind moet worden. Zodra het model voldoende getraind is, is het labelen in principe niet meer nodig.

Zodra een point cloud gelabeld is, is deze nog niet klaar om te trainen. Elk machine learning model verwacht een andere soort input van point clouds. In het geval van PointCNN betekent dit dat elke point cloud eerst opgesplitst moet worden in losse *.txt* bestanden van elke instance van een class. Deze bestanden kunnen vervolgens verwerkt worden tot 'batches' van punten. Van elke batch worden de punten en labels in aparte bestanden opgeslagen. Deze batches worden uiteindelijk allemaal toegevoegd in één bestand met behoud van de structuur van deze batches. Dit bestand kan vervolgens gebruikt kan worden voor het trainen van een PointCNN model.

De gelabelde data moet nog één stap ondergaan voordat deze klaar is voor het gebruik in een machine learning model. Hoewel elk machine learning model een andere datastructuur verwacht, is er een gemene deler. Vrijwel alle modellen verwachten de input data in *.h5* bestanden, deze hiërarchische bestanden kunnen data opslaan in verschillende lijsten in één bestand. Hiermee kan de input data opgeslagen worden in meerdere batches van een bepaald aantal punten in één bestand. Zodra de data op de juiste manier in het *.h5* bestand is geladen kan het gebruikt worden voor het trainen van het model.

5.2.2 Deelvraag 2: Hoe kan een machine learning model geïmplementeerd worden in een pipeline met als doel het automatiseren van de verwerking van point clouds?

Met de resultaten van deelvraag 1 is het mogelijk om een machine learning model te trainen, deze moet echter eerst geïmplementeerd worden. De implementatie van het model is een technische aangelegenheid en een tijdrovend proces. De bestanden waaruit het model bestaat moeten allereest in de pipeline geïmporteerd worden. Vaak zijn geavanceerde machine learning modellen zoals PointCNN op een abstracte manier gebouwd. Dit houdt in dat het model zo is opgebouwd dat het van zichzelf geen details over de implementatie bevat. Deze details worden bepaald door het model bepaalde waardes en parameters mee te geven bij de initialisatie. Hierbij wordt onder andere gebruik gemaakt van een bestand met daarin veel van de settings die het model gebruikt. Dit *model_settings* bestand bevat veel parameters die gebruikt worden door het model. Zo worden hierin onder andere de gegevens over de structuur van de input data en de instellingen voor het trainen opgeslagen.

Naast het bestand met de instellingen van het model moeten er ook placeholders geinitialiseerd worden. Deze lege variabelen hebben een bepaalde vorm of zijn van een bepaald type maar bevatten nog geen data. De placeholders worden gebruikt bij het trainen van het model om data op te slaan. Er worden onder andere placeholders aangemaakt voor het bijhouden van het aantal iteraties van het model en de batches waarmee het model traint. Daarnaast worden er ook *scalars* opgezet, deze variabelen kunnen statistieken van het model bijhouden tijdens het trainen. Scalars kunnen onder andere informatie opslaan over de accuracy en loss tijdens het trainen. Nadat de placeholders en scalars zijn geinitialiseerd moeten alleen nog een paar bestandslocaties van bijvoorbeeld het te exporteren getrainde model gekozen worden. Vervolgens kan er begonnen worden met het trainen.

Voor het uitvoeren van inference met een getraind model zijn vrijwel dezelfde stappen van toepassing. Hoewel er minder placeholders en scalars geinitialiseerd worden is het inladen van het model vergelijkbaar. Ook het importeren van een reeds getraind model is eenvoudig. Dit kan door simpelweg de bestandslocatie van een checkpoint van het getrainde model te specificeren.

5.2.3 Deelvraag 3: Hoe kunnen er met behulp van de geëxtraheerde objecten statistieken over het straatbeeld gevormd worden?

Met het getrainde machine learning model kan er inference op point clouds uitgevoerd worden. Dit wordt gedaan met de Inference-pipeline. Voor het uitvoeren van inference op een point cloud zijn minder stappen nodig dan voor het trainen van een mode. Allereerst hoeft de point cloud niet gelabeld te worden, het doel is immers het classificeren van de objecten in de point cloud op basis van wat het getrainde model heeft geleerd. De uitkomst van de inference is dan ook vergelijkbaar met een gelabelde point cloud. Voordat deze informatie hiervan gebruikt kan worden moet deze nog wel verwerkt worden. Vervolgens kunnen er op basis van het resultaat statistieken worden berekend.

De statistieken worden per class berekend, daarom worden eerst de punten van elke class opgesplitst in aparte arrays. Vervolgens kunnen de statistieken berekend worden op basis van de punten van elke class. Er ligt hier echter een uitdaging. Omdat de accuracy van de inference nooit 100% is, hierdoor zullen er altijd punten onjuist geclassificeerd zijn. Hierdoor kan het zo zijn dat de foutief geclassificeerde punten dicht bij het originele object, of juist op een volledig andere plek in de

point cloud liggen. Dit brengt de nodige problemen met zich mee. Er kan namelijk niet uitgegaan kan worden van het gegeven dat een groep punten per definitie een instance van een class is. Een groep punten die bijvoorbeeld foutief als prullenbak is geclassificeerd hoeft geen onderdeel te zijn van een prullenbak, terwijl dit misschien wel zo lijkt.

Er zijn twee manieren waarmee de statistieken berekend kunnen worden, triviaal en non-triviaal. Bij het berekenen van de statistieken op een triviale manier wordt zonder vooraf ingegeven informatie geprobeerd om de statistieken te berekenen. Deze methode is vatbaar voor het hierboven beschreven probleem omdat er geen manier is om effectief de groepen met foutief geclassificeerde punten op te sporen. Om dit soort onzekerheden enigszins te vermijden is het ook mogelijk om non-triviaal de statistieken te berekenen. Hierbij wordt gebruik gemaakt van vooraf ingestelde waardes die informatie bevatten over de ground truth van de te classificeren objecten. Zo kunnen bijvoorbeeld de afmetingen van prullenbakken gebruikt worden om te kijken of een groep punten die als prullenbak geclassificeerd is ook daadwerkelijk een prullenbak kan zijn. Op basis hiervan kunnen punten die bij voorbaat geen onderdeel van het geclassificeerde object kunnen zijn gefilterd worden. Zo is de kans groter dat de berekende statistieken daadwerkelijk juist zijn.

Ook is het mogelijk om zelf afmetingen van de geclassificeerde objecten te meten. De coordinaten van de point clouds doe voor dit project gebruikt zijn bevatten namelijk informatie over de afmetingen van objecten in de scans. Wanneer twee punten op bijvoorbeeld de x-as 1 meter van elkaar zijn verwijderd, zal het verschil tussen de x-coordinaten ook 1 zijn. Dit gegeven kan gebruikt worden om de afmetingen van de instances te berekenen.

Het succes van de statistiekenberekening is en blijft grotendeels afhankelijk van de prestaties van de inference. Toch is het mogelijk om door middel van slimme berekeningen bepaalde onzekerheden weg te halen.

6. Ontwerp

In de ontwerpfase lag de focus op het aanbrengen van structuur voor de op te leveren software en het proces daaromheen. In de onderzoeksfase is een ontwerp voor de softwareproducten opgezet. In dit hoofdstuk worden de resultaten en het proces van deze kortdurende fase besproken.

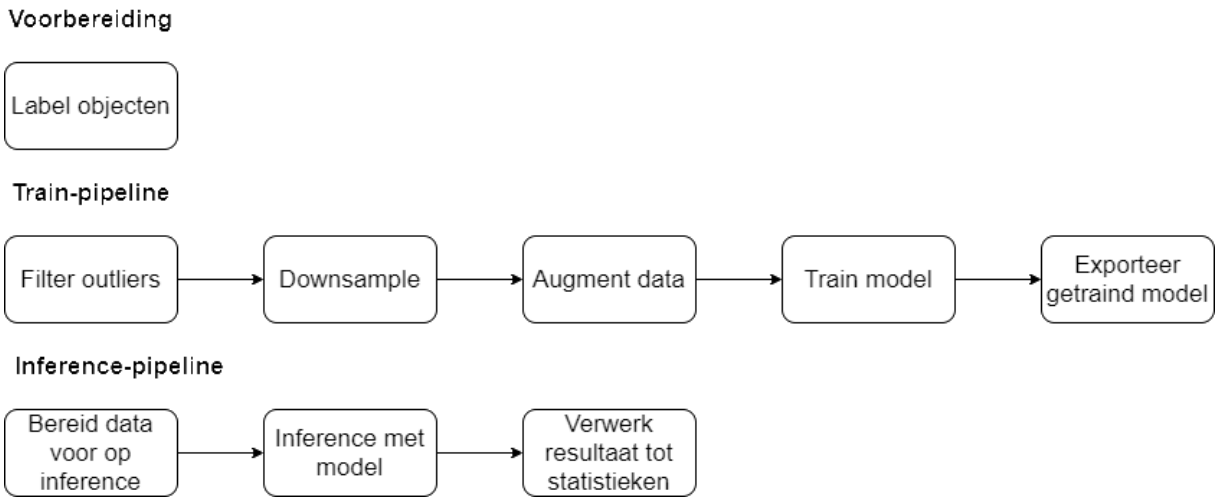
6.1 Software ontwerp

Het probleem is in twee deelproblemen opgesplitst omdat er twee aparte functies zijn die de software moest ondersteunen: trainen van een machine learning model en inference (gevolgtrekking) met een getraind machine learning model. Om dit te bewerkstelligen is er gekozen om twee softwareproducten op te zetten die elk een van deze functies op zich neemt. De softwareproducten zijn een aaneenschakeling van verschillende machine learning technieken om de data voor te bereiden en te verwerken. De aaneenschakeling van bovengenoemde technieken is een pipeline.

De twee softwareproducten zijn als volgt: de Train-pipeline en de Inference-pipeline. De pipeline die de point cloud kan verwerken tot nuttige informatie noemen we de Inference-pipeline. Deze Inference-pipeline kan met behulp van een machine learning model objecten in het straatbeeld herkennen. Om dit mogelijk te maken moet echter eerst het machine learning model in deze pipeline getraind worden om objecten in de point cloud te kunnen herkennen. Dit wordt gedaan met de Train-pipeline. Met het resultaat van de inference-pipeline kunnen statistieken berekend worden.

6.1.1 Train pipeline

De Train-pipeline zal met meerdere automatische stappen de point clouds verwerken zodat ze gebruikt kunnen worden om een machine learning model te trainen. Deze pipeline zal gebruik maken van vooraf gelabelde point clouds. De gelabelde data wordt verwerkt en voorbereid voor het trainen van de data. Hieronder valt onder andere het filteren van de outliers en het maken van augmentations van de data. Het model dat met deze pipeline getraind is kan vervolgens gebruikt worden in de Inference-pipeline.



Figuur 4 - Een overzicht van de pipelines met de stappen voor beide pipelines

6.1.2 Inference pipeline

De Inference-pipeline maakt gebruik van het machine learning model dat getraind is met de Train-pipeline. Het doel van deze pipeline is het verwerken van de ruwe point clouds tot nuttige informatie. Het reeds getrainde machine learning model wordt geïmporteerd en voorbereid voor de Inference. Daarnaast wordt ook de data voor de Inference ingeladen en worden eventuele bewerkingen hierop gedaan. Hierbij zal eerst de inference met het model plaatsvinden en vervolgens het resultaat hiervan verwerkt worden tot nuttige informatie.

7. Realisatie

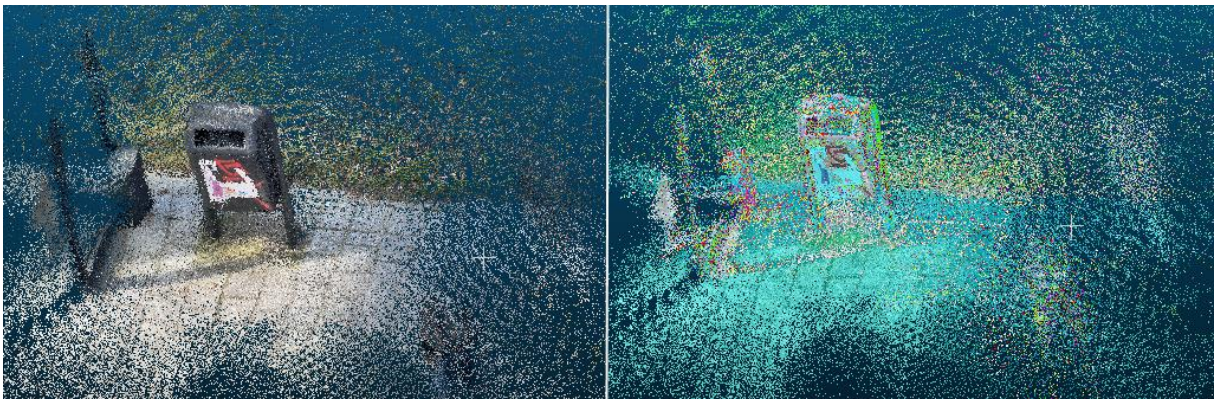
De realisatiefase is de langste fase van het afstudeertraject. In deze fase ligt de focus op de ontwikkeling van de software op basis van het ontwerp dat in de ontwerpfase is opgezet. Bij de start van de realisatiefase is gekozen om niet direct de focus te leggen op de ontwikkeling van de gehele pipelines, maar eerst een minimale aanvaardbare versie hiervan te bouwen. Deze worden Minimum Viable Products (MVPs) of baselines genoemd. Deze vroege versies van het eindproduct bevatten alleen de kern van de functies. Een baseline kan na afronding gebruikt worden als basis voor de rest van de pipeline. Voor de train-baseline houdt dit in dat er op basis van de beschikbare data een machine learning model getraind kan worden, de inference-baseline moet in staat zijn om inference uit te voeren op een point cloud. De baselines bieden een goede mogelijkheid om kennis te maken met de technische details en de implementatie van het PointCNN model. Naast dit hoofdstuk zijn de werkzaamheden en uitdagingen van de realisatiefase ook gedocumenteerd in het Realisatieverslag, deze is als bijlage bij dit document opgenomen.

7.1 Ontwikkelomgeving

In de analysefase van het afstudeertraject is gekozen om gebruik te maken van Jupyter Lab. Deze lokale ontwikkelomgeving is een tool waar de afstudeerder al ervaring mee heeft en is eenvoudig op te zetten. Het onderzoek in de analysefase en het begin van de ontwikkeling van de train-baseline zijn in deze lokale ontwikkelomgeving ontwikkeld. Zodra er voortgang werd gemaakt met de implementatie van het machine learning model bleken er echter hardware limitaties te zijn. Om het PointCNN model te compileren en te trainen dient het systeem te beschikken over een geschikte videokaart. Na enig onderzoek bleek Google Colab een oplossing te zijn. Deze online omgeving die vergelijkbaar is met de gebruikte lokale omgeving heeft toegang tot snelle processors en videokaarten van Google. De overstap naar deze omgeving heeft meerdere problemen met zich meegebracht. Zo zijn er veel aanpassingen gedaan worden om de online opslag van Google Colab te kunnen gebruiken. Daarnaast zijn niet alle libraries die in de lokale omgeving gebruikt zijn beschikbaar in Colab. Hierdoor kunnen bepaalde technieken die onderzocht zijn in de analysefase niet toegepast worden in de baseline. In het Realisatieverslag is meer informatie te vinden over de overstap van Jupyter Lab naar Google Colab en de uitdagingen die hierbij komen kijken.

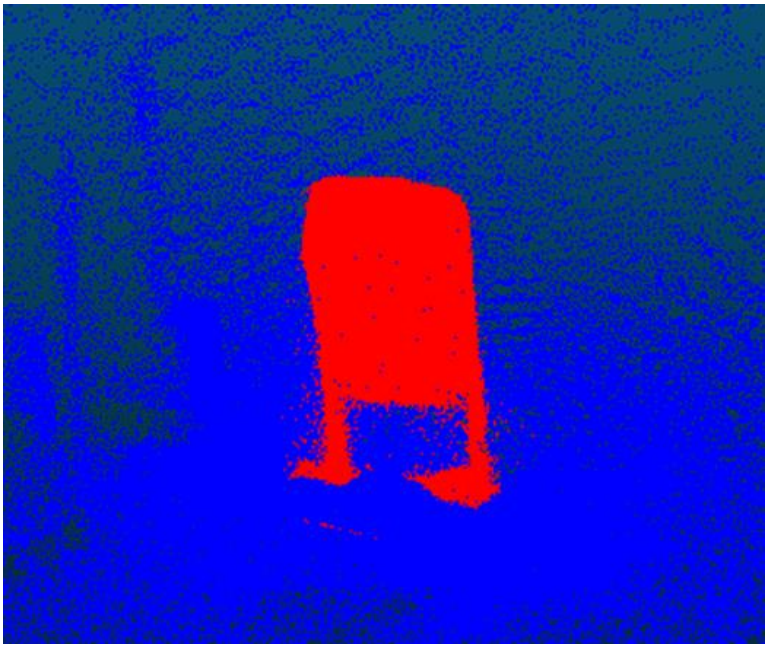
7.2 Labeling

Het labelen van de beschikbare data is eigenlijk de enige stap in de train-pipeline kan niet geautomatiseerd worden. Er zijn echter wel tools die dit proces eenvoudiger maken. Met een labeling tool is het mogelijk om groepen van punten te selecteren om die vervolgens toe te wijzen aan een bepaald label. Na onderzoek na verschillende tools is het oog gevallen op een open-source tool met veel positieve gebruikers. Het aanpassen van de beschikbare classes in de tool is eenvoudig te doen door het aanpassen van een. JSON-*bestand*. Bij het exporteren van de gelabelde bestanden deden zich echter problemen voor. Zo wordt de RGB-kleurinformatie van de point cloud onjuist opgeslagen en worden twee kolommen van de originele data overschreven door de label en instance kolommen van het labelen.



Figuur 5 - Een point cloud in kleurweergave vóór(links) en na het labelen.

In Figuur 5 is het gevolg van dit probleem te zien. De kleurinformatie van de point cloud is na het labelen onjuist. Om dit probleem te verhelpen zijn er aanpassingen gemaakt aan de labeling tool, dit is mogelijk omdat de tool open-source is. Na de aanpassingen is de gelabelde output bruikbaar voor verdere verwerking en gaat er geen data verloren.



Figuur 6 - De gelabelde punten van een afvalcontainer

De output van de labeling tool is een *.txt* bestand met daarin een 2-dimensionaal array met de punten van de point cloud. Deze punten hebben elk een coördinaat met X, Y en Z waarde, een R, G en B waarde voor de kleurinformatie en een waarde voor zowel de class als de instance.

```
10.550924301147461 -0.5300688147544861 59.77951431274414 206 192 160 0 -1
10.533666610717773 -0.5418468117713928 59.743770599365234 218 206 176 0 -1
11.448363304138184 0.019287100061774254 60.290802001953125 83 85 81 2 0
11.44347095489502 0.08564049750566483 60.287559509277344 68 70 68 2 0
11.44262981414795 0.14056630432605743 60.28813171386719 37 34 32 2 0
11.442754745483398 0.19066859781742096 60.28956604003906 34 32 30 2 0
```

Figuur 7 - De output van de label tool met van links naar rechts de X, Y, Z, R, G, B, Label en Instance waardes.

Zoals te zien in Figuur 7 hebben sommige objecten een instance waarde van -1, dit is het geval als de punten van dat label niet tot een object instance behoren. Voor de beschikbare dataset is dit alleen het geval bij het 'void' label, dit label bevat alle punten die niet als een andere class gelabeld zijn.

7.3 Train-baseline

De eerste baseline die ontwikkeld is tijdens de realisatiefase is de train-baseline. Deze baseline vormt de basis voor de train-pipeline. Tijdens de ontwikkeling van de train-baseline is er veel aandacht geweest voor het onderzoeken van de technische details en de implementatie van het model.

7.3.1 Preparatie van gelabelde data

De gelabelde data als output van de labeling tool is niet geschikt om direct te gebruiken voor het trainen van het PointCNN model. Elk machine learning model verwacht een andere soort structuur van de input data. Het PointCNN model verwacht een vrij overzichtelijke indeling waarbij de punten in batches van een veelvoud van 1024 punten in *.h5* bestanden opgeslagen zijn. De implementatie van het model op andere point cloud datasets maakt gebruik van batches van 8192. Daarom is er voor dit project ook gekozen voor batches van 8192. De beschikbare data bestaat uit hele straten met daarin meerdere objecten van verschillende classes. Per straat verwacht het model twee *.h5* bestanden met in elk bestand de verschillende batches. In de repository van het PointCNN model is informatie te vinden over het gebruik van het model met bepaalde veelgebruikte datasets. Als voorbeeld zijn onder andere scripts te vinden over de verwerking van bepaalde datasets naar de door PointCNN gevraagde structuur. Deze scripts zijn niet geschikt voor de beschikbare data van het afstudeerproject maar kunnen hiervoor wel aangepast worden. De scripts verwachten een specifieke structuur van data waarbij de punten van elke instance en label van de point cloud in een apart *.txt* bestand staan. Dit splitsen van de hele data naar class instances wordt gedaan door een onderdeel van de baseline.

Zodra de data is opgesplitst kunnen de aangepaste scripts uitgevoerd worden. De *.h5* bestanden met batches van data worden gegenereerd aan de hand van de opgesplitste bestanden en kunnen vervolgens worden gebruikt voor het trainen van het model. Voorafgaand aan het trainen wordt de data eerst gesplitst in een train- en validatiedataset.

7.3.3 Implementatie van PointCNN

Met de data in een bruikbare structuur kan het gebruikt worden voor gebruik met het model, deze moet echter eerst geïmplementeerd worden. In de repository van PointCNN is informatie te vinden over het gebruik van het model, dit is echter op basis van scripts. Voor het implementeren van het model in de baseline moet het model zelf ingeladen worden en moeten alle voorbereidende acties die hierbij horen aangepast worden aan de notebook.

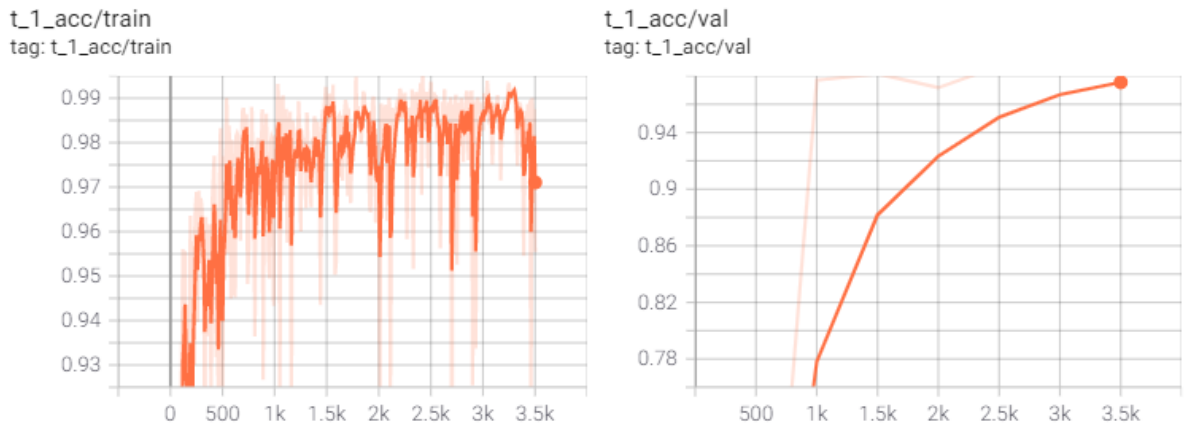
Het PointCNN model is opgebouwd op een abstracte wijze waardoor de details van het model bepaald worden door de implementatie ervan. Dit houdt in dat het model zelf bestaat uit python bestanden die op verschillende manieren ingezet kunnen worden. Zo is er geen definitieve structuur met de verschillende lagen van het model, maar wordt dit bepaald door de implementatie van het model. Dit wordt onder andere gedaan door middel van het “model_settings” bestand. Dit bestand bevat veel van de parameters die bij de initialisatie en uitvoering van het model worden gebruikt. Zowel het model als de settings kunnen als python bestand in de pipeline geïmporteerd worden alvorens geinitialiseerd te worden.

Voorafgaand aan de initialisatie worden er ‘placeholders’ opgezet, deze lege arrays of variabelen krijgen later op basis van de input data hun definitieve waardes. Deze placeholders zijn objecten die onderdeel van zijn van de Tensorflow library. Om deze placeholders hun waarden te geven moet het model geinitialiseerd worden met de data. Voorafgaand hieraan worden eerst nog een paar ingebouwde augmentations uitgevoerd, deze worden samen met de originele dataset gevoegd en aan het model meegegeven. Zodra het model is ingeladen moeten er alleen nog ‘scalars’ voor het bijhouden en meten van de prestaties opgezet worden. Scalars worden simpelweg gebruikt voor het wegschrijven van de prestaties van het model naar een samenvatting van de training. Vaak heeft een scalar als input een ‘tensor’, dit is een multidimensionaal array dat wordt gebruikt om data in het model rond te sturen en aan te passen. Zo zijn er scalars die de accuracy of de accuracy per class bijhouden aan de hand van de tensors die deze informatie bevatten. Vervolgens wordt de optimizer bepaald, deze probeert de snelheid en prestaties van het model tijdens het trainen te verbeteren. Als laatste worden de bestandslocaties van de te exporteren checkpoints van het model bepaald. Deze checkpoints bevatten de informatie van het getrainde model bij een bepaalde iteratie en kunnen in een later stadium geïmporteerd worden om mee verder te trainen of te gebruiken voor inference.

De implementatie van het model werd niet vereenvoudigt door het feit dat Google Colab slechts enkele versies van Tensorflow ondersteund, de Tensorflow versie waarop het PointCNN model is ontwikkeld zat hier niet bij. Hierdoor zijn er legio aanpassingen onder de motorkap van het model nodig geweest om verouderde methodes en variabelen van Tensorflow die niet meer in gebruik zijn aan te passen naar moderne alternatieven. Na deze aanpassingen kan het model op een enkele functie na geinitialiseerd en gebruikt worden. De alternatieven van andere Tensorflow versies verwachten andere parameters waardoor een aanpassing van de lagen van het model nodig zou zijn om dit werkend te krijgen. De impact van deze functie op de resultaten is lastig aan te duiden. In overleg met Appnormal is besloten om hier geen uitgebreide aandacht aan te besteden. Ook omdat het nog niet zeker is of het systeem in de toekomst op Google Colab gedraaid zal worden.

7.3.4 Trainen met het PointCNN model

Nadat het model geïmporteerd en geinitialiseerd is kan het gebruikt worden om te trainen met de klaargestoomde data. Het trainen vindt plaats in een for-loop die door de batches van training data heen loopt. Elke iteratie van deze loop wordt de batch met data ingelezen en wordt deze gebruikt om te trainen. Elke tiende iteratie van de loop wordt de loss, accuracy en accuracy per class berekend. Deze statistieken worden opgeslagen in de summary van het model. Elke vijfhonderd iteraties wordt een checkpoint van het model opgeslagen naar de vooraf ingestelde locatie. De tijdsduur van het trainen is afhankelijk van de grootte van de input data en de hoeveelheid iteraties.



Figuur 8 - De accuracy van het trainen(links) en de validatie van het trainen die gelijktijdig plaatsvindt

De summary van het model kan ingeladen in Tensorboard, een tool van Tensorflow waarmee de activiteiten van een model gevisualiseerd worden. In Tensorboard zijn vervolgens grafieken zoals van de accuracy in Figuur 1Figuur 8 te vinden. Deze kunnen zowel tijdens als na het trainen getoond worden.

7.4 Inference-baseline

Bij de ontwikkeling van de inference-baseline is gebleken dat dit een aanzienlijk kleiner project is dan de train-baseline. Dit komt niet alleen door het feit dat er minder acties nodig zijn voor het uitvoeren van inference ten opzichte van het trainen. Ook is er bij de ontwikkeling van de train-baseline al kennis vergaard over de implementatie van PointCNN.

7.4.1 Preparatie van inference data

Om data voor te bereiden op inference is het allereerst belangrijk om de twee soorten data waarmee inference kan plaatsvinden te onderscheiden. Voor inference kunnen zowel ruwe point clouds zonder label informatie als gelabelde point clouds gebruikt worden. Het voornaamste verschil tussen deze twee is dat de resultaten van de inference bij een gelabelde point cloud na afloop getoetst kunnen worden. Dit is mogelijk omdat de door inference geclassificeerde punten vergeleken kunnen worden met de gelabelde punten die als de ‘ground truth’ worden gezien. Bij deze evaluatie van het resultaat worden statistieken zoals de ‘accuracy’ en ‘intersection-over-union’ van de geclassificeerde data gemeten. In Hoofdstuk 8 worden deze waardes verder toegelicht.

Net zoals bij het trainen van het model, wordt ook voor inference de data verwacht als *.h5* files in dezelfde structuur als de geprepareerde files voor de train-baseline. Voor de gelabelde data kunnen dan ook de scripts van de train-baseline gebruikt worden om de data voor te bereiden. Omdat de baseline de minimaal aanvaardbare versie van de pipeline is, is er gekozen om voor de inference-baseline alleen te werken met gelabelde data als input. Het mogelijk maken van het gebruik van ruwe data in de inference-baseline is een toevoeging die gedaan wordt na afronding van de baseline.

7.4.2 Implementatie van PointCNN

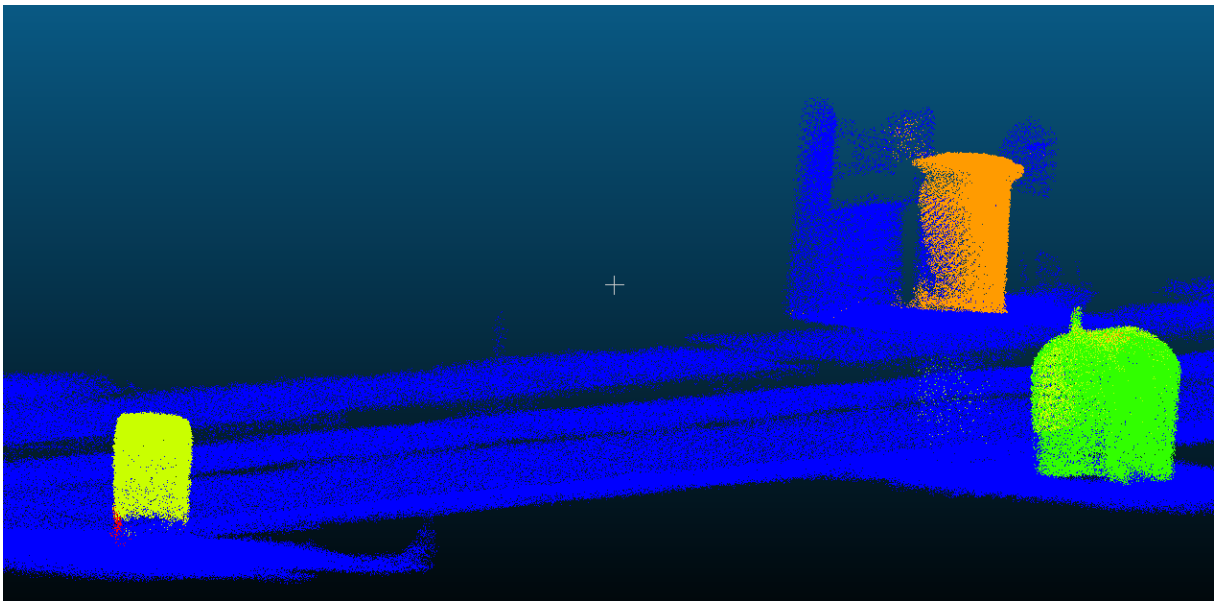
De implementatie van PointCNN in de inference-baseline is eenvoudiger dan de implementatie in de baseline. Dit komt mede door de opgedane kennis bij de ontwikkeling van de train-baseline. Het model wordt net zoals in de train-baseline ingeladen met de input data. Voor de inference hoeven er echter minder placeholders geinitialiseerd te worden. Tijdens de inference worden er namelijk

weinig statistieken bijgehouden zoals dat bij het trainen wel gebeurt. Het importeren van een getrainde versie van het model is eenvoudig. De bestandslocatie van het gewenste checkpoint van het getrainde model kan in de notebook worden aangegeven voor gebruik bij de inference.

Tijdens de inference wordt op eenzelfde manier als bij het trainen van het model de data per batch ingelezen en verwerkt. De *predictions* van het model worden bijgehouden en opgeslagen in een apart *.h5* bestand. Naast het geschatte label per punt in de point cloud wordt ook de zekerheid van de schatting opgeslagen.

7.4.3 Evaluatie

Nadat de inference heeft plaatsgevonden kan de data samengevoegd worden, dit wordt gedaan met behulp van python scripts. Net zoals de input point cloud zijn namelijk ook de resultaten van de inference in batches verdeeld. Naast het *mergen* van de data produceert dit script ook een bestand met daarin simpelweg de geschatte labels op volgorde van de punten in de originele point cloud. Dit bestand kan later worden gebruikt voor doeleinden zoals het visualiseren van de resultaten van de inference.



Figuur 9 - Een deel van het resultaat van de inference

Het evalueren van de inference data wordt gedaan door deze te vergelijken met de ground truth en vervolgens te tellen hoeveel van de punten juist en onjuist zijn geclassificeerd. Aan de hand hiervan kan de gemiddelde precisie van de classificatie gemeten worden. In de inference-baseline wordt het resultaat nog niet verwerkt tot statistieken zoals het aantal gedetecteerde instances van een bepaald object in de point cloud.

7.5 Train-pipeline

Op basis van de train-baseline is de train-pipeline ontwikkeld. Het verschil tussen de baseline en de pipeline zijn de toevoegingen op de baseline die de resultaten van het trainen verbeteren. Deze toevoegingen worden voornamelijk op de data uitgevoerd voordat deze gebruikt wordt om het model te trainen. Sommige van deze toevoegingen, zoals outlier removal en augmentations, zijn reeds onderzocht tijdens de analysefase. De implementatie hiervan wordt hieronder toegelicht.

7.5.1 Outlier removal

In de analysefase zijn er experimenten gedaan voor outlier removal met de Statistical Outlier Filter van de PCL library. Deze is echter niet beschikbaar in Google Colab en kan ook niet geïnstalleerd worden. Het alternatief voor deze library is gevonden in de vorm van Open3D. De Statistical Outlier Removal kan aangeroepen worden met een standaarddeviatie en hoeveelheid naburige punten als parameters. Deze functie returnt twee waardes: de point cloud zonder outliers en een lijst met de originele indices van de 'inliers'. Hiervan hebben we alleen de laatstgenoemde nodig. Met deze lijst van indices kunnen alle punten behalve de outliers uit de originele point cloud gehaald worden waardoor er op de outliers na geen informatie verloren gaat.

De optimale mate en daarmee de effectiviteit van outlier removal verschilt per dataset. Om bij benadering een goede set parameters voor de outlier removal te verkrijgen zijn er handmatige testen gedaan. Zo is er outlier removal op verschillende sterktes uitgevoerd en zijn deze resultaten vergeleken. Daarnaast zijn ook verschillende niveaus van agressiviteit van outlier removal getest. Dit is gedaan door het uitvoeren van inference met een model dat getraind is met verschillende outlier removal parameters. De parameters die de beste prestaties van het model oplevereden zijn gekozen als de definitieve parameters.

7.5.2 Augmentations

Zodra de outliers van de point clouds verwijderd zijn kunnen ze in principe gebruikt worden voor het trainen van het model. Er kunnen echter ook augmentations toegepast worden. Deze kleine veranderingen aan de originele point cloud kunnen gebruikt worden om de hoeveelheid training data aanzienlijk te vergroten. Tijdens de analysefase is er onderzoek gedaan naar noise augmentation, rotatie en scaling augmentation en downsampling. Bij noise augmentation en downsampling wordt de structuur van de point cloud niet gewaarborgd, bij rotatie en scaling augmentations wel. Deze data die deze augmentation technieken genereren worden naast de gewone data opgeslagen en gebruikt voor het trainen van het model. Hiermee wordt het model robuuster en kan het beter omgaan met point clouds van mindere kwaliteit. In het Realisatieverslag worden de gebruikte augmentations uitgebreid toegelicht met oog op de technische details.

Bij het implementeren van het model in de train-pipeline bleek dat er al methodes waren die augmentations uitvoeren op de dataset. Deze augmentations waren inbegrepen in een bestand vol functies die gebruikt worden voor het gebruiken van het model. De methodes gebruiken jitter/noise en rotation en scaling technieken op vrijwel dezelfde manier als in het Realisatieverslag toegelicht wordt. Hierbij is het grootste verschil dat de rotation en scaling gelijktijdig worden uitgevoerd en samen één verdubbeling van de originele dataset produceren. De downsampling augmentation zit hierbij echter niet inbegrepen. Er is gekozen om deze augmentation uit te voeren vlak ná de outlier removal, maar voordat de data verwerkt wordt tot *.h5* bestanden. Door deze downsampling augmentation ontstaat er een kopie van de originele data die gedownsampled is. Vervolgens kan de originele data en de gedownsamplede variant verwerkt worden tot *.h5* bestanden en gebruikt worden voor de verdere uitvoering van de Train-pipeline.

Om de effectiviteit van de augmentations te waarborgen is het van belang dat de parameters die gebruikt worden bij het augmenten geoptimaliseerd zijn voor de gebruikte dataset. Het is echter lastig om de optimale parameters voor alle verschillende vormen van augmentations te bepalen. Het is namelijk qua tijdsbestek niet mogelijk om systematisch in kleine stapjes door de parameters te

lopen en per stap het model opnieuw te trainen. Door de vele variabelen die invloed hebben op de prestaties van de point cloud naast de augmentations is het dan ook lastig om tot de perfecte parameters te komen. Door handmatig te testen met modellen getraind met verschillende augmentation parameters is een redelijk onderbouwde keuze voor de parameters gemaakt.

7.6 Inference-pipeline

Net als bij de train-pipeline zijn er ook toevoegingen gedaan aan de inference-baseline om zo tot de uiteindelijke inference-pipeline te komen. Het aantal toevoegingen aan de inference-pipeline is minder dan bij de train-pipeline omdat de inference-pipeline minder acties bevat.

7.6.1 Gebruik van ruwe data voor inference

De inference-baseline kan gebruik maken van gelabelde data die op dezelfde manier verwerkt wordt als in de train-pipeline. Het is ook mogelijk om ruwe data te gebruiken voor inference, deze ruwe data bevat geen informatie over de labels of objecten in de point cloud. Het is goed denkbaar dat de inference voornamelijk wordt gebruikt met ruwe data wanneer het eindproduct daadwerkelijk in gebruik wordt genomen. Het einddoel van inference is namelijk het vergaren van nieuwe informatie uit abstracte data, niet het verifiëren van de prestaties van het model zoals dat met gelabelde data kan.

Het verwerken van de gelabelde data is vrij eenvoudig, voor de ruwe data ligt die iets genuanceerder. Omdat deze data twee kolommen met data mist, namelijk de label en instance kolommen, kunnen de scripts van de train-pipeline hiervoor niet gebruikt worden. Deze scripts zijn op de ruwe data aangepast zodat de label en instance kolommen die ontbreken in de ruwe data niet worden meegenomen in de verwerking. Daarnaast zijn de onderdelen van de scripts die op deze kolommen waren gebaseerd aangepast zodat de scripts wel de gewenste output creëren. Om te zorgen dat het uitvoeren van de inference-pipeline goed gaat met zowel ruwe data als gelabelde data moet er in de pipeline onderscheid gemaakt worden tussen deze twee. Dit wordt gedaan met behulp van een boolean die aangepast kan worden op basis van het type input data. Afhankelijk van de staat van deze boolean worden scripts wel of niet uitgevoerd en worden acties zoals het verifiëren met behulp van gelabelde data al dan niet gedaan.

7.6.2 Outlier removal

Net zoals in de train-pipeline is outlier removal ook toegepast in de inference-pipeline. Het gebruik van deze functie kan bepaald worden in het *model_settings* bestand zodat de gekozen instelling voor beide pipelines wordt gebruikt. Een model dat is getraind op data zonder outliers presteert namelijk beter bij de inference als bij de input data ook de outliers zijn verwijderd. Het outlier removal algoritme gebruikt dezelfde methode als de outlier removal van de train-pipeline. De parameters die hiervoor worden gebruikt zijn ook gelijk aan die van de train-pipeline en worden opgeslagen in de *model_settings.*

7.7 Verwerking van inference resultaat tot statistieken

Voor het berekenen van statistieken op basis van de inference resultaten is het belangrijk om onderscheid te maken tussen twee manieren van het berekenen, triviaal en non-triviaal. Bij het berekenen van statistieken op een triviale manier wordt dit gedaan door puur naar de resultaten van de inference te kijken. Hierbij worden de statistieken zonder vooraf ingestelde waardes berekend. Het nadeel hiervan is dat het erg lastig is om hiermee goede resultaten te verkrijgen wanneer de

accuracy van de inference niet bijzonder hoog is. Door triviale berekening te gebruiken is het niet mogelijk om de bij voorbaat nutteloze informatie effectief weg te filteren.

Met non-triviale berekening kan dit wel enigszins. Hierbij worden waardes ingesteld waarmee bepaald kan worden of punten wel of niet bij de objecten in de point cloud horen. Zo kunnen bijvoorbeeld de afmetingen van een prullenbak gebruikt worden om te kijken of de als prullenbak geclassificeerde punten onderdeel uit kunnen maken van een prullenbak of niet. Een nadeel hiervan is wel dat de waardes die gebruikt worden voor deze methode afgestemd moeten worden op de dataset. Desalniettemin is het met deze non-triviale manier mogelijk om onzekerheden uit het resultaat te filteren en zo tot betere statistieken te komen.

7.7.1 Clustering

Ervan uitgaande dat het resultaat van de inference toereikend is, kan er gekeken worden naar manieren waarop de groepen van punten gesplitst worden. Clustering is een verzamelnaam voor verschillende technieken waarmee groepen van punten opgespoord kunnen worden in een point cloud. De definitie van een groep is in dit geval de punten die zodanig bij elkaar liggen dat ze meer bij de desbetreffende groep horen dan bij een andere groep.

Een populaire clustering techniek is k-means clustering. Hierbij worden met behulp van vectors de middelpunten van clusters gezocht en vervolgens gekeken welke punten bij welk cluster horen. Dit algoritme presteert het best op clusters van dezelfde grootte, dit zou echter geen groot probleem moeten vormen. Het is aannemelijk dat de geclassificeerde objecten van een label min of meer vergelijkbaar zijn in grootte. Een nadeel van deze functie is echter dat de meeste k-means clustering algoritmes een vooraf ingesteld aantal te zoeken clusters verwacht. Omdat het aantal objecten per straat zal verschillen is dit geen toereikend algoritme.

Een clustering algoritme dat wel toereikend kan zijn is ‘agglomerative clustering’. Deze methode heeft een recursieve aanpak waarbij elk punt in eerste instantie als losse cluster gezien wordt. Per stap worden de clusters die het meest gelijk aan elkaar zijn samengevoegd. Hierbij kan er gespeeld worden met de maximale afstand die punten in een cluster mogen hebbe om uiteindelijk een waarde te vinden die goed aansluit bij de resultaten van de dataset.

7.7.2 Foutief geclassificeerde punten

Zoals eerder benoemd is de accuracy van de resultaten niet geheel toereiken voor het berekenen van de statistieken. Vaak worden er punten onjuist als een bepaald label geclassificeerd waarbij het onderscheid tussen juist en onjuist geclassificeerde punten lastig te maken is. Er kunnen echter wel acties worden ondernomen om de hoeveelheid foutief geclassificeerde punten te verminderen. Zo bevat het inference resultaat ook waardes voor de ‘confidence’, dit is de zekerheid waarmee elk punt geclassificeerd is. Uit testen tijdens de realisatiefase is gebleken dat het verwijderen van punten met een confidence lager dan 98% helpt bij het opschonen van het resultaat.

Figuur 10 - Het inference resultaat van de 'afvalcontainer' class

In de afbeelding hierboven is duidelijk te zien hoe de foutief geclassificeerde punten een probleem kunnen vormen. De punten die zich linksboven en rond de afvalcontainer bevinden zijn foutief geclassificeerd maar kunnen moeilijk opgespoord worden. Deze punten liggen zo dicht bij elkaar dat zelfs een agressieve outlier removal niet voldoende is voor het verwijderen ervan. Ook het verwijderen van punten met een lage confidence heeft niet genoeg invloed. Ter illustratie: bij het verwijderen van alle punten met een confidence van minder dan 98% worden slechts drieduizend van de grofweg 120 duizend punten verwijderd. Tijdens de realisatiefase is dan ook gebleken dat het erg lastig is om de foutief geclassificeerde punten op te sporen en weg te filteren. Er kan gesteld worden dat het resultaat van de inference simpelweg nog niet toereikend genoeg is om met enige vorm van zekerheid de statistieken te berekenen. In hoofdstuk 8 worden de behaalde resultaten verder toegelicht en in hoofdstuk 10 worden aanbevelingen gedaan voor het verbeteren van zowel het inference resultaat als het berekenen van de statistieken.

8. Resultaten

De resultaten van de inference op basis van het getrainde model zijn gemeten met voor het model onbekende data. Dit houdt in dat de inference is uitgevoerd op data die het model nog nooit ‘gezien’ heeft. De voor inference gebruikte data is vergelijkbaar met de data waarop het model getraind is. Alle classes die in de train-dataset te vinden zijn, zijn ook in de inference-dataset aanwezig. De objecten van de classes komen echter uit nieuwe scans die gemaakt zijn op andere plekken dan de oude scans. Hiermee wordt een realistische situatie voor het gebruik van dit systeem nagebootst.

De prestaties van het model worden gemeten aan twee statistieken: de accuracy en de ‘intersection-over-union’ (IoU). De accuracy geeft simpelweg aan hoeveel procent van de punten juist zijn geclassificeerd. Deze waarde geeft echter niet altijd een goede weergave van de prestaties, daarom wordt er ook gekeken naar de IoU. Voor het meten van deze statistiek worden ‘bounding boxes’ gebruikt. Een bounding box is de kleinst mogelijke kubus die in de 3D ruimte om alle punten van een bepaald label getekend kan worden. Voor de intersection-over-union wordt vergeleken in hoeverre de bounding box van de ground truth overlapt met de bounding box van de geschatte waardes. Als de bounding boxes exact gelijk zijn is deze waarde 1.

Figuur 11 - Een deel van de straat waarmee de prestaties van de verschillende modellen vergeleken worden met twee prullenbakken, een kliko en een afvalcontainer.

De data waarop de inference wordt uitgevoerd bestaat uit één straat met daarin de te classificeren objecten. Sommige classes hebben meerdere instances, zoals prullenbakken en de kliko’s, en andere classes komen slechts één keer voor in deze straat.

8.1 Prestaties van de inference

Om de prestaties van het model in kaart te brengen zijn er vier varianten van het getrainde model vergeleken. Het gebruikte model en dataset is hierbij hetzelfde maar de pipeline waarmee het model is getraind is verschillend. De varianten die vergeleken worden hebben oplopend aantal toevoegingen. De meest rudimentaire variant is de baseline zoals die aan het begin van de realisatiefase is opgeleverd. De andere varianten hebben extra toevoegingen zoals augmentations en outlier removal, of combinaties hiervan.

De modellen die met elkaar vergeleken worden zijn als volgt: het model getraind door de baseline, het model getraind met augmentations, het model getraind met augmentations én downsampling en

het volledige model. Deze laatste variant is getraind met de opgeleverde versie van de pipeline en is het meest compleet. Dit model maakt gebruik van augmentations, downsampling en outlier removal.

Pipeline variant	Gemiddelde accuracy	Gemiddelde IoU
Baseline (zonder toevoegingen)	88.32%	0.27
Augmentations	88.77%	0.30
Augmentations + downsampling	89.79%	0.43
Augmentations + downsampling + outlier removal	92.22%	0.52

Figuur 12 - De gemiddelde accuracy en IoU van de varianten van de pipeline

In Figuur 12 is goed te zien welke impact de pipeline toevoegingen hebben op de prestaties van het getrainde model. Zo ligt de accuracy aanzienlijk hoger bij de volledige pipeline dan bij de baseline. Dat het model van de volledige pipeline een stuk beter presteert is ook te zien aan de verschillen in IoU tussen deze twee: de baseline heeft een IoU van 0.27 waar dat bij de volledige pipeline 0.52 is. Dit is bijna een verdubbeling van de mate waarin de bounding boxes overeenkomen met de ground truth. De outlier removal heeft hier een groot aandeel in, aangezien hierdoor punten die aan de buitenkant van de point cloud liggen weg gefilterd worden. Het effect van de outlier removal is goed terug te zien door de IoU's van het model zonder outlier removal en mét outlier removal te vergelijken. Ook de downsampling heeft een flink aandeel in het verhogen van de gemiddelde IoU.

Een handige manier om de prestaties van het model te visualiseren is met 'confusion matrices'. Dit zijn tabellen waarin de geschatte punten uitgezet worden tegen de ground truth. Bij een goede accuracy zal er een diagonale lijn van linksboven naar rechtsonder te zien zijn. Door het gebruik van confusion matrices kan goed in beeld worden gebracht welke labels succesvol geclassificeerd zijn en waar de foutief geclassificeerde labels zich bevinden.

Figuur 13 - De confusion matrices van verschillende varianten van het model op een volledige straat

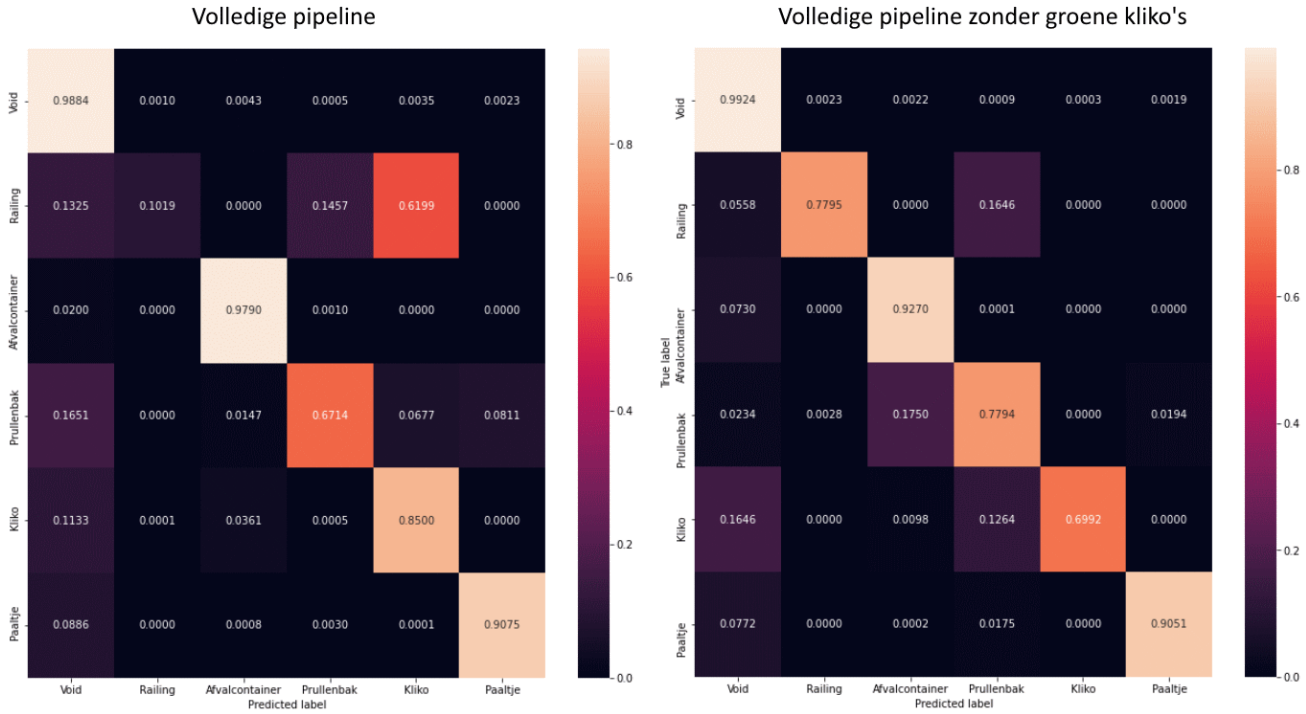
In Figuur 13 worden de confusion matrices van vier varianten van de pipeline getoond. De confusion matrix rechtsonder is van het model dat getraind is met de volledige train-pipeline met alle gerealiseerde toevoegingen. Er is een duidelijk verschil te zien tussen de matrix van de baseline en de matrix van het volledige model.

Een terugkerende achilleshiel van de bovenstaande modellen is het label van de kliko. Dit heeft echter een goede reden. Zoals eerder benoemd is de dataset van de training anders dan de dataset

die gebruikt is voor de inference. Waar de train-dataset alleen oranje-grijs gekleurde kliko's bevat, bevat de inference-dataset naast een oranje-grijs gekleurde kliko ook groene kliko's.

Figuur 14 - De kliko van de train-dataset (rechts) en de groene kliko's die onderdeel zijn van de inference-dataset

Door het verwijderen van deze groene kliko's in de inference-dataset wordt zowel de accuracy als de IoU flink verhoogd. Met een toename van 20% op de accuracy en 0.21 op de IoU is dit een ontzettende verbetering. Deze verbetering is goed verklaarbaar omdat het model niet getraind is op de groene kliko's.



Figuur 15 - De confusion matrix van de prestaties van inference op de inference-dataset zonder groene kliko's.

Wanneer deze resultaten in een confusion matrix worden gevisualiseerd zoals in Figuur 15 is er een duidelijke diagonale lijn zichtbaar. Het is goed te zien dat het model een stuk hoger scoort op de accuracy van de kliko class. Daarnaast heeft dit als gevolg dat ook andere classes zoals de railing aanzienlijk beter geclassificeerd wordt.

Pipeline variant	Gemiddelde accuracy	Gemiddelde IoU
Augmentations + downsampling + outlier removal	92.22%	0.52
Augmentations + downsampling + outlier removal (zonder groene kliko's)	97.23%	0.73

Figuur 16 - De gemiddelde accuracy en IoU van het model getest mét en zonder groene kliko's

De prestaties van de pipeline op de inference-dataset zonder groene kliko's zijn bevredigend. Dit is een aanduiding dat het goed mogelijk is om objecten uit het straatbeeld te classificeren met een machine learning systeem. Door het gebruik van een bredere en meer diverse train-dataset zou het resultaat nog verder verbeterd kunnen worden. Het resultaat van de volledige pipeline zonder de kliko's is vrij goed en kan gezien worden als gevolg van de succesvolle realisatie van de pipelines. Toch kunnen hier kanttekens bij geplaatst worden, de gemiddelde accuracy is van alle classes. Hierbij wordt ook het 'void' label meegenomen. Dit label bevat die punten die niet expliciet gelabeld zijn en dus ook niet interessant zijn voor het eindresultaat. Omdat dit label het gros van de punten van de point cloud bevat geeft dit een enigszins vertroebeld beeld van de daadwerkelijke nauwkeurigheid van de classificatie van de andere classes. De precisie van de best geclassificeerde class ná de void class is zo'n 70%, dit is de afvalcontainer class. Dit houdt in dat 30% van de punten die als afvalcontainer geclassificeerd zijn niet daadwerkelijk tot een afvalcontainer behoren. Elk geclassificeerd punt heeft naast het toegewezen label ook een zekerheid score. Deze score geeft echter niet altijd een goed beeld van of het punt foutief geclassificeerd is. In Hoofdstuk 10 worden aanbevelingen gedaan over het verbeteren van de prestaties van het model en het berekenen van de statistieken.

9. Conclusie

In dit hoofdstuk kan aan de hand van de reeds beantwoorde deelvragen de hoofdvraag beantwoord worden. Daarnaast worden de resultaten kort besproken.

9.1 Hoofdvraag: Hoe kan Machine Learning ingezet worden om voor gemeentes relevante objecten en statistieken uit het straatbeeld te extraheren uit point clouds?

Om met behulp van machine learning informatie uit point clouds te extraheren zijn er twee systemen nodig. Een systeem dat een machine learning model traint om bepaalde objecten in point clouds te herkennen en een systeem dat met dit getrainde model deze objecten in een point kan classificeren. Deze systemen zijn aaneenschakelingen van verschillende technieken die ook wel pipelines worden genoemd. Er zijn dan ook twee pipelines nodig: een Train-pipeline en een Inference-pipeline. Deze pipelines hebben als doel het trainen van een machine learning model en het gebruiken van dit getrainde model voor het verkrijgen van informatie uit point clouds.

Om objecten en statistieken uit point clouds van het straatbeeld te extraheren is het allereerst van belang dat het machine learning model wordt geleerd welke objecten als relevant worden gezien. Dit wordt gedaan met de Train-pipeline. Deze pipeline verwacht als input een point cloud waarvan de relevante objecten gelabeld zijn als zodanig. Bij dit labelen worden alle punten van de point cloud een waarde gegeven op basis van welk soort object ze belichamen. Daarnaast kan er hierbij ook rekening gehouden worden met objecten die vaker dan één keer voorkomen.

Gelabelde point clouds kunnen vervolgens worden gebruikt om het model te trainen. Voordat dit mogelijk is moeten deze echter nog verwerkt worden naar de datastructuur die het model verwacht. Ook kunnen er nog bewerkingen op de gelabelde data uitgevoerd worden om deze te optimaliseren voor het trainen van het model. Zo kunnen de outliers verwijderd worden, deze punten dragen niet bij aan de nuttige informatie die de point cloud belichaamt. Ook kunnen er augmentations op de gelabelde point clouds uitgevoerd worden. Deze kleine aanpassingen aan de point cloud kunnen gebruikt worden om de hoeveelheid beschikbare training data te vermenigvuldigen. Zodra de outliers verwijderd zijn en de augmentations zijn uitgevoerd kan de data omgezet worden naar de datastructuur dat geschikt is voor het model.

Voorafgaand aan het trainen moeten de nodige variabelen van het model geïnitialiseerd worden om vervolgens het model zelf in te laden en op te zetten met de vooraf bepaalde instellingen. Het trainen gebeurt aan de hand van de vooraf ingestelde hoeveelheid iteraties en grootte van de te verwerken point clouds. Na elke iteratie wordt een checkpoint van het model opgeslagen dat vervolgens gebruikt kan worden in de Inference-pipeline.

In de Inference-pipeline is het proces minder ingewikkeld. De point cloud waarop de inference moet plaatsvinden wordt ingeladen en de outliers worden verwijderd. Vervolgens wordt de point cloud omgezet naar de juiste datastructuur voor de inference. Het gewenste checkpoint van het getrainde model kan daarna geïmporteerd worden en samen met de rest van de benodigde variabelen geïnitialiseerd worden. De Inference kan hierna plaatsvinden, het resultaat hiervan wordt opgeslagen maar bestaat op dat moment uit allerlei gefragmenteerde delen van de originele point cloud. Deze fragmenten kunnen echter samengevoegd worden om zo de point cloud te krijgen die door middel van de inference gelabeld is.

Nadat de inference heeft plaatsgevonden is het mogelijk om op basis van de resultaten statistieken te verkrijgen. Hierbij is de precisie van de berekende statistieken sterk afhankelijk van de precisie van de inference. Logischerwijs kunnen de statistieken accurater berekend worden als het resultaat van de inference ook accurater is. Voor het berekenen van deze statistieken moeten eerst de foutief geclassificeerde punten zoveel mogelijk weg gefilterd worden. Bij een beter inference resultaat zal de focus minder hierop liggen en meer op de verschillende statistieken die te verkrijgen zijn. Om te zorgen dat de foutief geclassificeerde punten zoveel mogelijk worden weggewerkt kan onder andere outlier removal worden toegepast. Ook kan er gekeken worden naar de zekerheid waarmee elk punt geclassificeerd is als onderdeel van een bepaald label. Het berekenen van de hoeveelheid geclassificeerde punten kan hierna gedaan worden door het aantal groepen van punten te tellen. Dit kan onder andere gedaan worden door clustering methodes.

Op basis van de behaalde inference resultaten is gebleken dat het lastig is om de statistieken te berekenen. De accuracy is niet toereikend voor het effectief kunnen bepalen van de hoeveelheid instances van een class in de point cloud.

10.Aanbevelingen

Het doel van dit project is het onderzoeken of het mogelijk is om machine learning in te zetten om geautomatiseerd point clouds van het straatbeeld te verwerken tot nuttige informatie. Het eindproduct is dan ook niet een volledig gepolijst systeem te zijn. Wel biedt het opgeleverde systeem de handvatten om de pipelines te verbeteren en te verfijnen in de toekomst. Tijdens het afstudeertraject zijn er technieken onderzocht die uiteindelijk niet geïmplementeerd konden worden wegens tijdgebrek of doordat andere onderdelen meer prioriteit hadden. Deze mogelijke toevoegingen zijn in dit hoofdstuk samengevat. Daarnaast wordt er advies geleverd over hoe het systeem tot betere resultaten zou kunnen komen.

10.1 Beschikbare data

De beschikbare dataset bij aanvang van het project was niet optimaal. Zo was de dataset vrij beperkt waardoor er van sommige classes weinig diverse training data was. Hoewel augmentations dit voor een deel kunnen opvangen is het verstandig om in de toekomst met een bredere dataset te werken. Hierbij is het advies om van elke class meerdere instances te verkrijgen voorafgaand aan het trainen. Zo is de kans op overfitting van het model kleiner en wordt het model robuuster. Een goed voorbeeld hiervan zijn de resultaten van de afvalcontainer. Het model heeft moeite met het classificeren van afvalcontainers omdat het maar getraind is op één enkele afvalcontainer. Daar komt bij dat de het exemplaar in de beschikbare dataset afwijkt van dat van de test data omdat het object van waarop getraind is beplakt is met stickers. Het trainen met meer diverse data kan een hogere accuracy opleveren voor dit soort objecten. Dit heeft als voordeel dat niet alleen het object zelf beter wordt geclassificeerd, maar ook andere objecten niet onterecht als dat object worden geclassificeerd.

10.2 Train-pipeline

De prestatiewinst in de train-pipeline is voornamelijk te halen op het gebied van de augmentations, outlier removal en de instellingen van het trainen. Voor de parameters van zowel de augmentations als de outlier removal is er nog ruimte voor verbetering door middel van het optimaliseren op basis van de beschikbare dataset. Toch zullen verdere optimalisaties van de huidige augmentations en outlier removal slechts een bescheiden verbetering opleveren. Er kan wellicht meer winst behaald worden door nieuwe technieken toe te voegen aan de pipeline. Zo kan er gekeken worden naar andere vormen van augmentations. Een combinatie van voxelgrid downsampling met jitter kan als een unieke extra vorm van augmentation gebruikt kunnen worden. Hiermee ontstaat er een point cloud waarvan de punten gelijk zijn verdeeld over de ruimte maar wel onregelmatig is door de jitter. Een ander potentieel interessante techniek is ground removal. Hierbij worden alle punten onder een bepaald niveau verwijderd. Dit zou het aantal void punten flink kunnen verminderen. Het void label wordt bij de inference vaak goed geclassificeerd, de winst op gebied van de resultaten zal daarom niet heel groot zijn. Het grootste voordeel zit waarschijnlijk in de grootte van de point clouds. Door ground removal zal dit aanzienlijk slinken wat snellere training en inference als gevolg heeft. Daarnaast is het aandeel van interessante punten in de point cloud ook groter na het weghalen van de grond.

Er kan ook er winst behaald worden door het aanpassen van de (hyper)parameters van het model. Door het aanpassen van bijvoorbeeld het aantal iteraties, de ‘learning rate’ of de gebruikte optimizer zou het model beter kunnen gaan presteren. Gelieerd hieraan is overfitting, door het optimaliseren

van deze instellingen kan dit eventueel beter voorkomen worden met een robuuster model als gevolg.

Zoals in hoofdstuk 7.3.3 wordt benoemd is een van de functies van het PointCNN niet in gebruik door de incompatibiliteit van Google Colab met bepaalde Tensorflow versies. Hoewel het niet mogelijk is om te zeggen hoe goed te prestaties van het model zijn wanneer deze functie wel in gebruik is, is het waarschijnlijk verstandig om hier onderzoek naar te doen. Het oplossen van dit probleem zou voor een stijging in precisie moeten zorgen. Indien de pipelines in de toekomst buiten Google Colab gebruikt zullen worden zal dit een vrij eenvoudige taak zijn.

10.3 Inference-pipeline

Aan de inference-pipeline kunnen vrij weinig echte verbeteringen worden gemaakt, dit komt vooral omdat het aantal acties van deze pipeline kleiner is dan van de train-pipeline. Toch kan net zoals bij de train-pipeline de outlier removal verbeterd worden. Daarnaast kunnen ook de parameters die het model gebruikt voor het uitvoeren van de inference geoptimaliseerd worden.

Een andere mogelijke toevoeging aan deze pipeline is het gebruik van lijsten van bestanden. Op het moment kan er óf een bestandslocatie van een ruwe point cloud, óf een lijstje met de verwerkte gelabelde bestanden gebruikt worden. Indien het proces gestroomlijnd kan worden en alleen gebruik maakt van een lijst van bestanden voor de inference zou dit een verbetering zijn. Deze lijst kan vervolgens ook gebruikt worden voor het samenvoegen van de resultaten en het al dan niet evalueren van de prestaties. In de huidige inference-pipeline moeten de bestandslocaties van de inference resultaten nog handmatig worden gekozen.

10.4 Verwerking tot statistieken

Zoals onder andere beschreven staat in hoofdstuk 7 blijkt het lastig te zijn om accurate statistieken te extraheren uit het resultaat van de inference. De aanbevelingen om dit proces te verbeteren kunnen opgesplitst worden in twee categorieën. Er kan onderscheid gemaakt worden tussen nieuwe soorten statistieken en manieren om het verzamelen van statistieken te verbeteren.

10.4.1 Verbeteren van de berekening van statistieken

Het verbeteren van de berekende statistieken is een kwestie die niet op zichzelf staat. Dit proces kan namelijk aanzienlijk vereenvoudigd worden wanneer de precisie van het inference resultaat hoger is. De aanbevelingen op het gebied van statistieken zijn dan ook voornamelijk gericht op het omzeilen van de foutief geclassificeerde punten. Bij een label met een accuracy van boven de 90% zou slechts een goed gekalibreerde outlier removal al voldoende kunnen zijn. Zolang de accuracy hieronder zit zullen er meer technieken gebruikt moeten worden.

Zoals in het hoofdstuk over de realisatie wordt besproken kunnen de groepen van punten opgespoord worden met clustering. Hierbij is in dit project gekeken naar ‘k-means clustering’ en ‘hierarchichal clustering’. Er zijn echter verschillende algoritmes voor deze vorm van clustering die nog niet onderzocht zijn. Daarnaast zijn er ook nog andere methodes van clustering die wellicht tot betere resultaten kunnen leiden. Deze methodes kunnen in de toekomst onderzocht worden.

Een van de aanbevelingen is het gebruiken van een tweede machine learning model dat bijvoorbeeld gebruik maakt van part-segmentation om de resultaten per label te scheiden. Het is verrassend lastig om meerdere groepen van punten van elkaar te scheiden op basis van de afstand tussen deze

groepen. Een model dat getraind is om deze groepen met segmentation uit elkaar te houden zou hiervoor een innovatieve oplossing kunnen zijn.

Een andere innovatieve manier is het gebruik van image recognition. Deze machine learning techniek kan objecten ontdekken in 2D afbeeldingen. Indien er automatisch afbeeldingen van het inference resultaat gemaakt kunnen worden, is dit geen slechte oplossing. Het image recognition model kan getraind worden op afbeeldingen van de geclassificeerde objecten met kleurinformatie en deze vervolgens zelf herkennen in het inference resultaat. Het verkrijgen van accurate resultaten met image recognition is over het algemeen eenvoudiger dan met 3D data zoals point clouds. Zodra de image recognition de juiste objecten gevonden heeft moeten deze echter nog wel gekoppeld worden aan de bijbehorende punten in de point cloud. Dit kan wellicht gedaan worden door met behulp van een raster de afbeeldingen te verkrijgen waarbij elke afbeelding een bepaald deel van het raster behandeld. Aan de hand van de plek in het raster kan vervolgens de locatie in de point cloud gekoppeld worden.

Bovenstaande verbeteringen zijn technisch uitdagend en een aanzienlijke hoeveelheid werk om te realiseren. Vooralsnog is het advies om eerst te focussen op het verbeteren van de prestaties van de inference voordat deze technieken worden gebruikt.

10.4.2 Andere soorten statistieken

Bij het afstudeerproject is alleen gefocust op het berekenen van de hoeveelheid instances van een class in het resultaat van de inference. Naast deze statistiek zijn er nog andere statistieken te berekenen die interessant kunnen zijn voor de eindgebruiker. Zo kunnen de afmetingen van de geclassificeerde objecten bepaald worden aan de hand van de coordinaten in de point cloud. Het nut hiervan is echter discutabel wanneer er gebruik wordt gemaakt van de non-triviale methode van het berekenen van de statistieken.

Een statistiek die waarschijnlijk nuttiger is voor het eindgebruik is de locatie van de geclassificeerde objecten in de point cloud. Op basis van de beschikbare data van het afstudeerproject is dit echter niet mogelijk. Om dit te kunnen bewerkstelligen moet er namelijk gps-data bij de point clouds meegeleverd worden. In overleg met Appnormal is besloten om dit buiten beschouwing te laten tijdens het afstudeerproject. Het gebruiken en implementeren hiervan is echt een project op zichzelf en heeft aardig wat voeten in de aarde.

11. Discussie

Het doel van dit project is de ontwikkeling van een systeem dat op basis van machine learning point clouds informatie en statistieken uit het straatbeeld kan verkrijgen. Gezien de beschikbare data en de tijd die ervoor stond kan gesteld worden dat het project vrij succesvol is afgerond. Daarnaast is het project zo opgebouwd en gedocumenteerd dat eventuele toekomstige uitbreidingen eenvoudig toe te voegen zijn. De resultaten van de inference zijn redelijk toereikend en geven een goede indicatie van of deze techniek ingezet kan worden voor het beoogde doeleinde van Appnormal. Daarnaast is er advies uitgebracht over de voortzetting van het project en de verbeterpunten van het opgeleverde systeem.

Toch bleek het lastig om met de behaalde resultaten de statistieken van geclassificeerde objecten te berekenen. Hiervoor is de gemiddelde accuracy van de geclassificeerde objecten te laag. De werkzaamheden bij het berekenen van de statistieken waren daarom vooral gericht op het opsporen en verwijderen van de foutief geclassificeerde punten en niet op het berekenen zelf. De aanbevelingen over de statistieken in hoofdstuk 10 kunnen pas effectief gebruikt worden zodra de resultaten echt toereikend zijn.

Dit hogere resultaat kan bewerkstelligd worden door het gebruik van een grotere en meer diverse dataset bij het trainen. Ook kunnen optimalisaties van de gebruikte technieken in de pipelines zorgen voor een verbetering van het resultaat. Deze verbetering is echter klein in vergelijking met de mogelijke verbetering van het gebruik van een grotere dataset.

Het systeem kan gebruikt worden om objecten in het straatbeeld te herkennen, het berekenen van statistieken over deze objecten is echter nog niet mogelijk. Het einddoel is daarom deels bereikt. Het is mogelijk om met een scan van een onbekende straat objecten in deze straat te vinden, het is echter niet direct mogelijk om hier ook statistieken aan te verbinden.

Het afstudeerproject vormde een interessante, innovatieve en bovenal uitdagende opdracht waarbij op voorhand niet zeker was wat de uitkomst zou zijn. Het systeem kan gebruikt worden als goede basis om op verder te ontwikkelen en daadwerkelijk in gebruik te nemen.

Bibliografie

Classification, Part segmentation en semantic segmentation. (2017). [Illustratie]. https://github.com/charlesq34/pointnet/blob/master/doc/teaser.png

Hu, Q. (z.d.). QingyongHu/RandLA-Net. GitHub. Van https://github.com/QingyongHu/RandLA-NetQi

Huang, Qiangui, Wang, Weiyue, Nuemann, & Ulrich. (z.d.). qianguih/RSNet. GitHub. Van https://github.com/qianguih/RSNet

Qi, R., C., Su, Hao, Mo, Kaichun, & Guibas. (z.d.). charlesq34/PointNet. GitHub. van https://github.com/charlesq34/pointnet

Qi, R., C., Yi, Li, Su, Hao, Guibas, & J, L. (z.d.). charlesq34/pointnet2. GitHub. van https://github.com/charlesq34/pointnet2

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (z.d.). yangyanli/PointCNN. GitHub. Van https://github.com/yangyanli/PointCNN

Mandrioli, D. (z.d.). Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor. GitHub. Geraadpleegd op 28 mei 2021, van https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor

Armeni, I., Sax, A., Zamir, A. R., & Savarese, S. (2016). S3DIS dataset. Stanford EDU. http://buildingparser.stanford.edu/dataset.html

ShapeNET. (2021). ShapeNet. https://shapenet.org/

Bijlagen

Bijlage 1 – Realisatieverslag

Het realisatieverslag is als losstaand document meegeleverd in het afstudeerdossier.