Selvkørende Biler

2. semester - Forår 2019

Eksamensgruppenr. og hus: S1924791396 - Hus B
Projekt(arbejds)titel: Selvkørende biler og teknologien bag
Gruppens medlemmer: Andreas Work - 66712 - awork@ruc.dk Louise Lundegaard - 66459 - loulund@ruc.dk Mathias Ravn Stephansen - 66453- mravns@ruc.dk Thomas Mortensen - 66437 - thomort@ruc.dk Thor Kjøller Hartmann - 66391 - thkjha@ruc.dk Tobias Dolma Eichler - 66392 - tde@ruc.dk
Vejleder: Kim Sandholdt
Dato: 27. maj 2019
Antal tegn: 129.111 (inkl. illustrationer) **Antal normalsider:** 53,8

Abstract

This paper seeks to study how the different technological artefacts in a self-driving car function. In addition, it also investigates how one can learn about the Artificial Intelligence used in these cars, by constructing a virtual 3D simulation using the engine Unity. Some of the more relevant technological artefacts are explained, as well as their link and connection to one another, to get a sense of the coherence between the different sensors and the AI in a self-driving car. The process of learning by constructing a simulation is examined by using John Dewey's theory about "Learning by doing", as well as Donald A. Schön's concept of reflective practice. A discussion focuses on the learning process and discusses whether "Learning by doing" or the traditional and more literary methods contributes mostly to the learning about self-driving cars. This paper concludes that one can learn about complex topics, such as Artificial Intelligence, by using either the traditional learning methods or by using "Learning by doing" separately, however it is more efficient if the two methods are both used together.

Indholdsfortegnelse

1. Indledning

1.1 Motivation

Motivationen for dette projekt munder ud i en fælles interesse for at forstå og arbejde med komplekse teknologiske systemer. Det interessante tager udgangspunkt i, hvordan vi som studerende kan være med til at udvikle eller undersøge nye teknologier eller forbedre eksisterende teknologier. Vi har alle bidt mærke i, at der de seneste år er blevet snakket meget om selvkørende biler både fra virksomhedernes og politisk side. Vi synes derfor det kunne være spændende at undersøge, hvordan selvkørende biler fungerer, ved at lave en simulation i Unity, som er et open source udviklingsværktøj, designet til 3D visualiseringer.

1.2 Problemfelt

"Wonder is the beginning of all wisdom". Sådan lyder det fra den gamle og anerkendte græske filosof Sokrates omkring 400-450 f.Kr. Selvom citatet stammer for så mange år siden, beskriver det begyndelsen på dette projekt rimeligt nøjagtigt. Projektet tager udgangspunkt i en undren om, hvordan selvkørende biler fungerer og en lyst til at finde svar, som til sidst resulterer i ny viden. Men hvordan finder vi bedst frem til denne nye viden? Det er det spørgsmål vi stiller gennem rapporten og til os selv. Det kan umiddelbart lyde simpelt, men vi ønsker ikke kun at læse os frem til den nye viden, men derimod også forsøge at genskabe denne viden ved at arbejde med det. Det bidrager forhåbentligt til en bedre og mere dybdegående forståelse. Måden, hvorpå vi vil gøre dette, er ved at lave en simulering af en selvkørende bil, som skal ende ud i at være så tæt på virkeligheden som muligt. Ved at opbygge simuleringen lærer vi gennem selve processen og udarbejdelsen af simuleringen, samt det gør det nemmere at videreformidle den viden vi opnår. Ideen om en selvkørende bil er ikke ny, men med de mange teknologiske fremskridt inden for Artificial Intelligence, sensorer og software er selvkørende biler tæt på at blive en realitet. Det er teknologier, som er meget komplicerede, men som er til rådighed. Vi mener, at det teknologiske stadie selvkørende biler er i nu, gør at selvkørende biler er en spændende og relevant problemstilling at tage fat i. Problemformulering og arbejdsspørgsmålene tager udgangspunkt i det ovenstående, hvilket munder ud i følgende.

1.3 Problemformulering

Hvordan kan vi gennem konstruktionen af en simulation undersøge og forstå AI og udvalgte teknologiske artefakter i en selvkørende bil?

1.4 Arbejdsspørgsmål

- Hvordan har udviklingen af selvkørende biler udfoldet sig igennem tiden?
- Hvordan fungerer de teknologiske artefakter i en selvkørende bil og hvilken funktionalitet har de i det teknologiske system?
- Hvordan bliver Artificial Intelligence brugt i en selvkørende bil?
- På hvilken måde har konstruktionen af en simulering bidraget til vores forståelse og læring af AI?

1.5 Produktoversigt

I det nedenstående afsnit introduceres den selvkørende bil, som vil blive simuleret. Simulationen vil blive dybere beskrevet senere i projektrapporten. Grunden til at produktet introduceres så tidligt i rapport, er for at give læseren en baggrundsviden om hvad projektet handler om og hvad der bliver arbejdet hen imod i forhold til de teoretiske afsnit, samt trin analysen af en selvkørende bil. Dette afsnit vil hjælpe til en bedre forståelse af blandt andet teoriafsnittet og hvorfor de valgte teorier er relevante. I udarbejdelsen af vores simulation er det ikke alle teorier, som bliver taget i brug, da vi ikke besidder den tekniske kunnen eller tid til at udarbejde så omfattende en simulation. Med dette i mente så vil alle teorierne være relevante, når der er tale om en selvkørende bil.

Simuleringen af en selvkørende bil vil bestå af tre hoveddele. Disse tre dele kan ses på vores visualisering i bilag 1.

Første del er de sensorer, som bliver simuleret på bilen og kan ses på punkt 1 i bilag 1. I dette scenarie vil denne sensor være en LiDAR-sensor, som blandt andet vil blive beskrevet i afsnit 5.2.1.1 omkring LiDAR.

Den anden del vil bestå af selve den selvkørende bils software, hvilket er punkt 2 i bilag 1, som igen vil blive defineret i detaljer senere. Dog skal det pointeres, at softwaren vil tage udgangspunkt i teori omhandlende et Artificial Neural Network og Machine Learning som begreb. Hvad denne software har til opgave, er at modtage nogle informationer fra de førnævnte sensorer og derefter lave en beslutning, omhandlende kørslen, ud fra denne selvsamme information.

Den sidste del, som simulationen vil bestå af, er selve bilen. Det er denne, som vil bevæge sig rundt på banen, i takt med at den modtager informationer, som den udfører handlinger ud fra. Disse handlinger er i dette tilfælde hvilken retning og hastighed bilen skal køre. Dette ses i bilag 1, punkt 3.

2. Teoriafsnit

I dette afsnit vil vi inddrage teoretiker og filosof John Dewey og hans læringsteori *'learning by doing'*, som skal være med til forklare og argumentere for den læring, vi har fået gennem konstruktionen af en simulation. Derudover ønsker vi at inddrage relevant teori, som belyser Artificial Intelligence i en selvkørende bil og de forskellige måder man kan træne Artificial Intelligence på. De forskellige teorier vil komme til udtryk i analyse af henholdsvis de teknologiske artefakter i en selvkørende bil, samt vores læring gennem dette projekt.

2.1 Learning by doing

For at forstå hvordan konstruktionen af en selvkørende bil i en simulation er med til at gøre os klogere på hvordan en selvkørende bil fungerer, er vi nødt til at undersøge hvad learning by doing går ud på og hvilke argumenter der ligger til grund for denne læringsteori. Derfor har vi valgt at inddrage John Deweys teori *'Learning by doing'*, som handler om, hvordan man lærer gennem erfaringer.

Learning by doing, også kendt som erfaringspædagogik eller konstruktiv læring, stammer helt tilbage til midten af 1900-tallet. En af pionererne inden for denne læringsteori er John Dewey. Han var en af skaberne og fortalerne for learning by doing, hvis hovedværk var bogen Democracy and Education (Larsen, 2007). Learning by doing tager afstand fra den

traditionelle læringsmetode, hvor studerende udelukkende skal have boglig eller mundtlig læring.

Formålet med learning by doing er, at viden skal skabes ved at danne erfaringer med det man arbejder med, samt at man selv skal prøve at lave det for at få det bedst mulige udbytte. Det skal forstås på den måde, at ny læring og viden ikke bare er noget som kan opstå på et øjeblik, men er resultatet af en dynamisk og længerevarende proces der er kommet til udtryk gennem en undren og en lyst til at finde svar (Ejlskov, 2019). John Dewey fortæller blandt andet, at for at læring skal være mest effektivt, bliver det stof, en studerende skal lære, nødt til at blive præsenteret på en måde, hvor den studerende kan relatere til det og på den måde få en bedre forståelse for det "*If knowledge comes from the impressions made upon us by natural objects, it is impossible to procure knowledge without the use of objects which impress the mind*" (Reed & Widger, 2008; Kapitel 20, l. 61 - 62). John Dewey havde en optimistisk tiltro til det enkelte menneskes egen læring og evne til at lede sit eget liv. Ved at lave sin egen selvdannelse, kan man udvikle evnen til at se kritisk og finde ud af hvordan man kan transformere nyt stof, med hensigt på at den nye læring bliver en integreret del af personligheden (Kolstrup, 2002; s. 12 - 13). Dewey banede på sin vis vejen til den deltagerstyrede undervisning, hvor der skal være et sammenspil mellem individet og vilkårenes omgivelser. *"For udviklingen af en sådan livsform er skolen et afgørende omdrejningspunkt, der ilter til undersøgelser og erfaringer"* (Kolstrup, 2002; s. 19), her nævnes det, at skolen er et af bedste instanser at indføre sådan en læring, da rammerne er til det. Dewey læringsteorier har også sat sit aftryk på skole praksisserne i dag, hvor projektarbejde og elevcentrering i høj grad er baseret på learning by doing af Dewey (Larsen, 2007). John Dewey forklarer ikke specifikt, hvad det er, der sker i selve læringssituationen, men det gør andre teoretikere, som i høj grad trækker på John Dewey, heriblandt Donald Schön. Schön har præsenteret verdenen for refleksion praksis, hvori han fremlægger tre måder, man reflekterer på.

Den første måde er *'knowing in action'*. Her benyttes viden som vi allerede har fået lært gennem tidligere erfaringer, som så gør, at man opfører sig på en bestemt måde i en bestemt situation. Det bliver også tit refereret til som tavs viden, det er viden vi benytter, men ikke reflekterer over eller kommer til udtryk, mens vi gør tingene (Schön, 2008; s. 63). Det er en refleksionspraksis, som kun til dels kommer til udtryk gennem vores learning by doing

proces. Et godt eksempel, der illustrerer dette, er når man cykler på en cykel. Her tænker vi ikke over hvordan vi skal gøre det, men vi gør det bare gennem tidligere erfaringer.

Det næste punkt er *'Reflection in action'*. Her reflekterer man over de ting, man gør mens man gør dem eller er i situationen. Schön beskriver det således:

> *"Usually reflection on knowing-in-action goes together with reflection on the stuff at hand. There is some puzzling, or troubling, or interesting phenomenon with which the individual is trying to deal. As he tries to make sense of it, he also reflect on the understandings which have been implicit in his action, understandings which he surfaces, criticizes, restructures, and embodies in further action."* (Schön, 2008; s. 64).

Det tydeliggøres her, at knowing in action og reflection in action går rigtig godt i flæng med hinanden. I reflection in action når man, til modsætning af knowing in action, at reflektere over hvad det er man skal gøre, og hvorfor man gør det. Denne refleksion sker meget hurtigt og hænger oftest sammen med hvilket resultat, der vil komme ud af ens endelige handling. Er man i tvivl om hvorvidt resultatet vil blive tilfredsstillende, overraskende eller uønsket, vil man ty til reflection in action (Schön, 2008; s. 70). Schön kommer selv med et eksempel, som omhandler en baseball spiller, der skal kaste en bold. Målet er at skyde forbi modstanderens batter. Her vil pitcheren overveje om han skal kaste oppe, nede, ud mod siden og om der skal være skrue i bolden. Pitcheren vil reflektere over tidligere handlinger han har lavet, og dermed komme frem til en beslutning med en forhåbning om, at det vil virke (Schön, 2008; s. 69).

Schön argumenterer også for, at det ovenstående eksempel kan karakteriseres som *'reflection on action'*, som er den sidste og mest anvendte refleksion praksis i vores projekt (Schön, 2008, s. 69). Her reflekterer man først over ens handlinger, efter man har lavet dem. Hvordan kan man gøre det bedre, hvad var det som gik godt, samt hvilke punkter kan forbedres. Ved at lave denne refleksion giver det mulighed for at finde ud af hvilke handlingsmåder, man skal lave fremover (Schön, 2008; s. 69). Det er nødvendigt for at udlede mening fra ens oplevelser. Eksempel på denne refleksionspraksis, kunne være en lærer, der var kommet hjem fra arbejde og reflekterede over hvordan hendes undervisning havde været. Her kan læreren så finde frem til andre mulige handlinger eller tiltag hun kan lave eller inddrage i undervisningen.

De teknologiske fremskridt har gjort det muligt at lave learn by doing i nogle felter, hvor man ikke har kunne gøre det på samme måde som før, især for studerende. Som nævnt ovenstående, er formålet med dette projekt at lære om selvkørende biler gennem konstruktionen af en simulation. Mere præcist at lære hvordan Artificial Intelligence fungerer i en selvkørende bil. Andre teknologiske artefakter vil vi finde kendskab til gennem brug af metoderne og relevant litteratur. I den ideelle verden, ville vi lave en rigtig selvkørende bil eller en mindre prototype, men det har vi hverken de nødvendige ressourcer eller tid nok til. Men der får vi igennem Unity mulighed for at konstruere og arbejde med den Artificial Intelligence, der bliver brugt i en selvkørende bil. På det grundlag har vi valgt at lave en simulation, da det giver god mulighed for at praktisere learning by doing teorien, i og med at simulation er en imitation af virkeligheden.

Opsamling

I det ovenstående afsnit fik vi redegjort for John Deweys læringsteori *Learning by doing*. Her argumenterer Dewey for at viden i høj grad er baseret på erfaringer ved at arbejde med det. Derudover blev der redegjort for Schöns tre tilgangsmåder til teorien. Den første, 'knowing in action', handler om den viden og de erfaringer man har på emnet inden man går i gang med emnet. Den næste tilgangsmåde, 'reflection in action', fokuserer på, at man har en forventning til det kommende resultat, i det man reflekterer over de ting som gøres under handlingen. Den sidste tilgangsmåde, 'reflection on action', er den mest brugte af de tre tilgangsmåder. Gennem 'reflection on action' reflekteres der over tidligere handlinger eller situationer, så der kan findes eventuelle fejl eller mulige forbedringer og løsninger.
Udover disse tilgangsmåder til learning by doing, har vi nu, i denne teknologiske tidsalder, mulighed for at gøre brug af begrebet learning by doing på nye måder. Simulering er et af de vidundere, som tillader at arbejde med og lære om emner, som ellers kan være svære at komme til.

2.2 Artificial Intelligence

I dette afsnit vil vi først forsøge at forklare hvad Artificial Intelligence egentlig er, inden vi vil komme ind på Artificial Neural Network og Machine Learning, som begge er former for Artificial Intelligence, vi bruger til vores simulation af en selvkørende bil. AI foregår i bilens software, som er punkt 2 i bilag 1.

Artificial Intelligence (AI) betyder direkte oversat kunstig intelligens. Der findes talrige definitioner og bud på hvad AI er, og vi har her valgt at inddrage fire overordnede idéer om hvad AI er for en størrelse, som vi har taget fra bogen "*Modern Artificial Intelligence: A Modern Approach*". På figuren nedenfor er der præsenteret fire forskellige definitioner på AI, med uddrag fra otte forskellige bøger.

Thinking Humanly	**Thinking Rationally**
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figure 1.1 Some definitions of artificial intelligence, organized into four categories.

(Russell & Norvig, 2010; s. 2)

Acting Humanly

Den første måde at definere AI, er en maskine som opfører sig menneskelig. Maskinen skal kunne udføre noget, som man normalt ser som en menneskelig adfærd. I forbindelse med dette er det relevant at kigge på Turing-testen. Dette er en anerkendt test, som har til formål at vurdere hvorvidt en robot/maskine opfører sig menneskeligt (Russell & Norvig, 2010; s. 2). Testen foregår gennem en skriftlig samtale, hvor en "forhørsleder" stiller en række spørgsmål. Hvis forhørslederen ikke kan kende forskel på om det er en maskine eller et menneske han skriver med, har maskinen "bestået" testen (Russell & Norvig, 2010; s. 2). For at bestå testen er der en række evner, som maskinen er nødt til at besidde. Først skal maskinen kunne føre en flydende samtale, såsom korrekt grammatik og korrekt brug af ord. Dernæst skal den kunne gemme informationer, som maskinen har på forhånd, eller som den får gennem samtalen. Når maskinen kan opbevare disse informationer, så må den også være i stand til at bruge disse informationer korrekt i en sammenhæng, der giver mening, eksempelvis til at besvare spørgsmål. Til sidst er maskinen nødt til at bruge Machine

Learning (som bliver uddybet i afsnit 2.2.1), for at kunne tilpasse sig nye omstændigheder og genkende nye mønstre (Russell & Norvig, 2010; s. 2).

Turing-testen tager ikke udgangspunkt i den fysiske fremtræden, idet *"physical simulation of a person is unnecessary for intelligence"* (Russell & Norvig, 2010; s. 3).
Der findes dog også en "total Turing-test", hvor det fysiske aspekt også spiller en lille rolle, i form af billedgenkendelse og evnen til at håndtere objekter og flytte på dem (Russell & Norvig, 2010; s. 3).

Thinking Humanly

I stedet for at opføre sig menneskeligt, er en anden tilgang til forståelsen af AI, at AI betyder en maskine, som tænker som et menneske. Denne tilgang kan være mere avanceret end de andre, idet det ville kræve, at man kender til den menneskelige måde at tænke på, noget som ikke er lige til at finde svaret på (Russell & Norvig, 2010; s. 3). I 2010, Russel og Norvig er der beskrevet tre forskellige måder at opnå denne viden på. Enten skal man forsøge at fange sine egne tanker, lave psykologiske eksperimenter, hvor man observere en person eller også skal man observere selve hjerne i aktion (Russell & Norvig, 2010; s. 3). Disse metoder vil vi ikke gå i dybden med i denne projektrapport, men kognitiv forskning er et stort område og man forstår stadig ikke 100% hvordan den menneskelige hjerne fungerer.

Thinking Rationally

Nogle vil definere AI som en rationel tankegang. Denne definition vægter det logiske højt og den perfekte AI vil altid løse et problem på den mest logiske måde. AI'en vil have en lang række logiske regler, som den bruger til at tage beslutninger. Antallet af regler vil være noget nær uendeligt og det vil derfor være sværere at finde en løsning på et problem jo flere faktorer der spiller ind. Problemerne, der kan opstå ved dette, er blandt andet, at det ikke altid er muligt at "oversætte" uformel viden til klare logiske regler. Dertil kan det kræve meget computerkraft i sidste ende, hvis der kommer situationer, hvor mange faktorer spiller ind (Russell & Norvig, 2010; s. 4).

Acting Rationally

Den sidste af de fire definitioner er, at en AI altid skal handle rationelt. Her vil maskinen tage den beslutning med det bedste resultat. Denne tilgang kræver først og fremmest et regelsæt, ligesom den fornævnte rationelle tankegang, men da der ikke altid er en tydelig "rigtig" ting

at gøre, så skal maskinen stadig kunne tage en beslutning i situationer, hvor der ikke er tydelige regler (Russell & Norvig, 2010; s. 4). Eksempelvis ville man kunne inkorporere færdselsregler, såsom hastighedsgrænser og vigepligter, men man kan ikke inkorporere specifikke regler for alle potentielle situationer. Derfor skal AI'en selv kunne finde en løsning til en uforudset hændelse.

Fordelen ved denne tilgang til AI er, at det er mere generelt end den rationelle tankegang, i det maskinen også kan tage beslutninger i situationer, hvor der ikke nødvendigvis er en decideret korrekt løsning (Russell & Norvig, 2010; s. 4 - 5).

Opsamling

I dette afsnit blev der redegjort for hvad Artificial Intelligence er, gennem fire kendte tilgange til at beskrive AI.

Den første definition på hvad AI er, beskrives ved "acting humanly". Her skal AI'en opføre sig på menneskelig vis. Dette kan afprøves med Turing-testen, hvor AI'en skal overbevise et menneske, gennem en samtale på skrift, om at AI'en er en rigtig person. Den gør dette ved at gøre brug af grammatiske regler og se mønstre i samtalen. Hvis mennesket ikke kan gennemskue AI'en, har AI'en bestået testen.

Anden definition på hvad AI er, beskrives ved "thinking humanly". Denne definition beskriver AI'en som en maskine, der kan formå at tænke som et menneske. Dette er mere avanceret end de andre tilgange, eftersom det ikke vides hvordan et menneske tænker. Men Russel og Norvig beskriver tre måder at opnå denne viden på. Disse tre måder beskriver; at forsøge at fange sine egne tanker, lave psykologiske, observerende eksperimenter, eller gennem observation af selve hjerne i aktion.

Den tredje definition beskriver, hvordan AI'en tænker logisk, via definitionen "Thinking rationally", hvilket er den tilgang, som den vægter højest, når den skal løse et problem. AI'en har en vis mængde regler at gøre brug af til at løse et givent problem. Des større et problem, des vanskeligere får AI'en, ved et problem med mange faktorer, og ikke alle problemer kan omskrives til logiske regler.

Den fjerde og sidste definition, på hvad en AI er for en størrelse, beskrives gennem definitionen "acting rationally". AI'en vil på denne måde altid vælge den tilgang, som giver

det bedste resultat. Eftersom der ikke altid findes et korrekt svar, skal AI'en selv kunne vurdere den bedste måde at løse problemet på.

2.2.1 Machine Learning

Vi har valgt at benytte os af Machine Learning til at lave vores selvkørende bil. Det er en måde at programmere på, hvor målet er, at computeren selv skal kunne lære sig frem til en løsning, ud fra den data man giver den, eller som den selv finder frem til. Har man arbejdet med en smule programmering, er noget af det første man lærer, at en computer ikke foretager sig noget, med mindre man beder den om det og hvis man vil have den til at udføre det korrekt, skal man skrive det helt ned til mindste detalje. Men ved brug af Machine Learning kan der opsættes nogle scenarier i form af mønstre, som den skal lære at genkende, og på den måde "tænke selv".

Når vi skal designe en bil, som selv skal kunne køre og eksempelvis ikke ramme ind i noget, vil det være for indviklet at designe alle de små ting, som den skal gøre i de forskellige situationer med "if-statements". If-statements er en kommando i programmering, som fungerer ved at fortælle computeren, at hvis X sker, så udfør en specifik handling. Eksempelvis hvis lyset er rødt, så stop ved lysreguleringen. Alternativet til disse, tæt på uendeligt mange if-statements det ville kræve, er så at benytte Machine Learning.

Der er flere forskellige måder, man kan lave Machine Learning på. Ud fra de mange kategorier, som findes af Machine Learning, har vi valgt kun at fokusere på supervised learning, unsupervised learning og reinforced learning, da vi finder dem mest relevante (Flach, 2012; s. 1 - 5).

2.2.1.1 Unsupervised Learning

Unsupervised learning er en underkategori i Machine Learning. Unsupervised learning er en metode, hvor computeren selv skal finde frem til løsningen ud fra nogle input, i form af en mængde data, som ikke er kategoriseret. Dermed kan den selv finde et mønster i de data, uden at vi fortæller den hvad den information er. Det er især smart, når computeren skal finde nye mønstre i data, som den ikke allerede ved er der.

Det kunne eksempelvis være to genstande computeren skal adskille fra hinanden. Den ene genstand kunne være et æble og det andet en appelsin. Ud fra unsupervised learning ville de

inputs omhandle forskellighederne af de to frugter, ned til mindste detalje. Computeren har en mængde data at kunne gøre brug af og med den data er det meningen, at den skal kunne karakterisere og separere de to frugter (Wilson & Keil, 1999; s. 858).

På baggrund af, at den selv skaber sine egne mønstre, kan det få systemet til at virke mere menneskeligt. Hvis systemet betragter menneskers handlinger, vil den efterligne de handlinger.

En ulempe med unsupervised learning er, at den selv udfører handlingerne, uden hjælp fra nogen. Det kan virke paradoksalt, eftersom det er formålet. Men det betyder, at computeren er sværere at kontrollere, når den selv finder frem til de ting som den vælger, og det kan samtidigt tage lang tid før det virker ordentligt, eftersom den skal køre en masse tests for indsamle nok data (Flach, 2012; s. 14 - 15).

2.2.1.2 Supervised Learning

En anden Machine Learning metode er supervised learning. Det virker lidt som unsupervised learning, på den måde, at computeren selv skal finde en fremgangsmåde til at undersøge data. Supervised learning leder efter mønstre i form af algoritmer, som passer til de informationer, i form af input, som computeren er blevet givet for at skabe generelle hypoteser til et emne der undersøges (Kotsiantis, 2007; s. 249). Det skal forstås som en simpel model, der formår at skelne mellem forskellige kategorier, med det formål at kunne forudsige hændelser og udfald. Kategorierne bliver brugt i forsøg (algoritmer), hvor computeren kan bruge dem til at forudsige udfald, den tager altså noget ikke kategoriseret data, og sammenligner med de allerede definerede kategorier, for derefter at kunne kategorisere disse data korrekt.

Med supervised learning får vi et resultat eller nogle data i forskellige kategorier, hvor vi ville have den til at komme frem til noget specifikt ud fra de bestemte inputs, som den er blevet givet.

Det kunne eksempelvis være e-mails, som den skal arbejde med, hvor den skal kategorisere e-mails enten som spam eller ønsket e-mails. Et kendt eksempel på spam, er e-mails med titlen "Nigeriansk prins har brug for din hjælp". Modtager man en e-mail med denne titel eller lignende, så er det formentlig spam. En måde man kan få den til undersøge det på er, at man vælger hvilke områder den skal holde øje med, eksempelvis ord eller sætninger, men det kan også være andre ting. I dette tilfælde fortæller vi derfor computeren, at det er spam, også

finder den ord eller sætninger som går igen i lignende spam e-mails. Efter den så har gennemgået dem gentagende gange, vil den så selv kunne finde frem til hvilke e-mails som er spam og hvilke som ikke er (Flach, 2012; s. 16 - 18).

Ulempen ved supervised learning er, at den ikke virker ligeså menneskelig som unsupervised learning. Supervised learning handler kun ud fra de områder, som den kender, og er blevet trænet i. Det betyder, at den ikke kan lære at tage hensyn til de samme ting som mennesker kan, med mindre den, som nævnt ovenstående, er blevet trænet i det. På baggrund af dette kan der ligge en masse forarbejde i at træne systemet til at handle menneskeligt, og der er større risiko for, at man kommer til at udelade vigtige elementer.

2.2.1.3 Reinforced Learning

Reinforced learning går ud på, at computeren selv skal løse en opgave og des tættere den er på at løse opgaven, des flere points får den. Den vil så prøve gentagne gange at få så mange points som muligt. AI´en starter med at bevæge sig tilfældigt rundt, nærmest i blinde, og hen ad vejen opdager computeren så, at nogle handlinger giver flere point i forhold til andre. Efter den så har kørt sekvensen mange gange, vil den begynde at finde ud af, hvordan den skal køre sekvensen uden at begå alt for mange fejltagelser og derfor opnår flest mulige points (Flach, 2012; s. 360 - 362).

Et af de vigtigste punkter ved at forstå reinforced learning er, at efter computeren vælger en handling, vil den blive (eller ikke) givet en "belønning", og fortalt hvad den efterfølgende handling har betydet. Computeren bliver til gengæld ikke fortalt hvilken handling, der ville have været bedst på længere sigt. Det er nødvendigt for computeren selv at erfare de forskellige situationer, så den selv kan begynde at virke optimalt (Kaelbling, Littman & Moore, 1996; s. 238).

Opsamling

Vi har i ovenstående afsnit introduceret og redegjort for Machine Learning og herunder unsupervised, supervised og reinforced learning.

Ved brug af Machine Learning menes der et program som selv er i stand til at genkende mønstre og handle ud fra de mønstre, som får det til at se ud som at den kan tænke selv. Disse mønstre lærer computer ud fra forskellige metoder, som hver har sine ulemper og fordele.

Ved brug af unsupervised learning, som er en af disse metoder af Machine Learning, som man kan bruge til at træne computeren til at tænke selv, giver man computeren nogle data, der ikke er kategoriseret i forvejen. Computeren bruger denne data den er blevet givet, til at lede efter mønstre, og dermed lære og forstå de nye inputs den modtager. Det positive ved denne tilgang er, at computeren kan betragte mennesker og lære gennem dem, og på den måde virke mere menneskelig. Det negative er, at det ikke er til at vide hvordan computeren tænker og derfor ikke kan spotte eventuelle fejl i programmet.

En anden metode inden for Machine Learning er supervised learning. I modsætning til unsupervised learning, bliver supervised learning givet nogle data, der allerede er kategoriseret, som den kan genkende og gøre brug af. Disse data er en slags regler for programmet, som vil prøve at skabe forbindelser og ligheder med nye inputs, som den modtager. I modsætning til unsupervised learning, lærer supervised learning ikke hen ad vejen. Den kan kun løse de problemer den er blevet trænet i. Fordelen ved dette er, at det altid er muligt at spotte fejl, og dermed rette dem. Men ulempen for at træne computeren i denne form for Machine Learning er, at der skal bruges lang tid på at opsætte regler for den og at den ikke virker nær så menneskelig.

Den sidste metode er reinforced learning. Computeren har nogle data at gøre brug af og skabe mønstre med, for så derved at løse et problem. Hver gang at computeren gør noget rigtigt, bliver den "tildelt" en "belønning", og omvendt hvis den gør noget forkert. Hver gang den gør noget forkert må den justere sit regelsæt, så den ikke begår samme fejl igen.

2.2.2 Artificial Neural Network

Artificial Neural Network (ANN) er en måde, hvorpå en computer kan tænke. ANN er designet ud fra hvordan en biologisk hjerne fungerer, og hvordan den indhenter information (Picton, 2000; s. 1). Men hvorfor er dette relevant, når der bliver talt om selvkørende biler? For det første er verden meget utilregnelig, og der er ingen garanti for at bilisten i næste vognbane ikke kører ind i dig. En AI-styret bil skal altså kunne forudse disse mulige scenarier og handle efter dem korrekt. Den måde AI'en gør dette på bedst, er ved at "tænke" som et menneske.

For at sætte dette teoriafsnit i kontekst med dette projekt, så er ANN'et selve hjernen i vores selvkørende bil. Det er denne som får nogle informationer af vores sensorer og derefter laver

en beslutning om hvorvidt bilen skal køre til højre, venstre, frem eller tilbage. ANN er altså en del af punkt 2 på vores visualisering af en selvkørende bil, som ses i bilag 1.

Først skal vi definere hvordan et menneske tænker. Mennesker behøver ikke at have set alle typer biler der nogensinde er lavet, før vi kan fortælle nogen, at det er en bil. Efter vi har set et X antal biler, kan vi begynde at sætte dem i en kategori der eksempelvis hedder 'biler'. For at noget kommer ind under denne kategori, skal den måske have fire hjul og et rat, men dette eksempel kommer vi tilbage til (Picton, 2000; s. 1 - 4). Disse kategorier er hvad vi bruger til at vurdere, om hvorvidt vi eksempelvis kigger på en bil eller en cykel.

Nu, hvor vi overordnet har defineret måden et menneske tænker på, kan vi begynde at sætte det ind i en kontekst for et ANN. Da et traditionelt ANN er opstillet på samme måde som en biologisk hjerne, fungerer den på samme måde som en menneskelig hjerne, til en vis grad. Et ANNs formål er at producere et output, når den får præsenteret et input (Picton, 2000; s. 1). Hvis vi går tilbage til eksemplet med biler; inputtet kan være en vilkårlig bil mens outputtet ville være AI'en, som fortæller, at det er en bil.

2.2.2.1 Regelbaseret system

Et Artificial Neural Network lyder meget let, og kunne måske gøres ved en simpel, men lang kode. Denne måde er også blevet afprøvet ved denne metode, som man kalder et regelbaseret system (Picton, 2000; s. ix - x). Et regelbaseret system er hvor man som programmør går ind og manuelt skriver en kode, som siger "Hvis den har 4 hjul, er 3 meter lang, er sort…, så er inputtet en ford sMax model 3". Denne metode kunne jo i teorien godt virke, og den har samme funktion som et ANN har. AI'en får et input og giver derefter et output om hvilken bil den registrerer. Dog er denne måde at lave AI på meget ustabil, det ville kræve et særdeles langt programmeringsarbejde, og derudover kunne den have en tendens til ikke at fungere nær så godt som et ANN. Ved brug af denne metode ville den have så mange forskellige variabler og forskellige output muligheder, at man simpelthen ikke ville have computerkraft nok til at udregne denne hver gang den ville få ny information ind (Picton, 2000; s. ix - 3).

Et andet problem med dette ville også være, at den umuligt ville kunne kategorisere alle typer biler perfekt hver gang. Denne metode at lave AI på har ingen generalisering og vil derfor ikke kunne finde den rigtige kategori med mindre alle dens variabler passer perfekt med den. Så hvad sker der hvis en person har malet sin bil blå i stedet for dens normale sorte farve? Dette ville måske ikke betyde så meget for en selvkørende bils beslutninger, men forstil dig

nu hvis det ikke var en bil? Tænk, hvis der var tale om et barn, eller en voksen mand, der står ved vejkanten.

2.2.2.2 Generalisering

ANN skal, som beskrevet tidligere, kunne få et input og give et output tilbage med den korrekte information. Mere teknisk, finder den mønstre den kan genkende og kategorisere. Hvordan kan man forbedre denne måde at genkende og kategorisere, på en måde der er bedre end hvad vi lige har omtalt? ANN er i stand til at lære og generalisere (Picton, 2000; s. 4 - 10).

En af de vigtigste og ikoniske egenskaber ved et ANN er dets evne til at generalisere. Det, at den kan generalisere, kan både spare computerkraft og beregningstid når den skal give et output. I stedet for at den kigger på 1000 forskellige variabler, når den kigger på en bil, så kan den kigge på meget mindre ved brug af generalisering (Picton, 2000; s. 4).

Hvad er generalisering? Et ANN kan under læringsprocessen lære, hvad der er relevant i forhold til at vurdere hvilken kategori den skal sætte dens input under. Eksempelvis hvis computeren har set 100 biler og fået fortalt, at de alle er biler. Dermed kan den udlede "en bil har altså fire hjul og et rat". Det er hermed ikke vigtigt om den er sort eller blå. Dette er et meget primitivt eksempel og i realiteten ville den måske have flere 100 forskellige variabler, som gør den i stand til at kunne fortælle, at det er en bil. Men hvad gør den, hvis en bil mangler et hjul? Det er her hvor generalisering kommer i brug. Et ANN er i stand til at give alle dens forskellige informationer vægte sammen med at den kategorisere bilen (Picton, 2000; s. 15 - 19).

Under læringsprocessen ser den alle biler har fire hjul på nær en bil. Den kan derfor sige, at hvis køretøjet den modtager som input har fire hjul, er der en stor chance for det er en bil. Med andre ord, så vægter den fire hjul højt i forhold til om det er en bil (Picton, 2000; s. 23 - 24).

Lad os prøve at sætte nogle hypotetiske tal på dette. Hvis vi forestiller os at computeren får præsenteret et billede af en bil som den skal kategorisere; hvis det er en bil, ville dens output være +1 og hvis det ikke er en bil, så vil dens output være -1. Computerens output er det som den samlede vægt af alle variablerne ender tættest på. Vi starter på 0, hvor den kigger på den første egenskab på billedet.

Bilen har 2 forsæder. Dette er vigtigt, men ikke altafgørende; +0.2.

Bilen har kun 3 hjul. Dette er meget vigtigt, og kan være afgørende; -0.7.

Bilen har 2 forlygter; +0.3.

Sådan fortsætter mønsteret. Summen af alle disse forskellige egenskaber hos bilen bliver til sidste lagt sammen, og hvis summen er tættest på -1 så er det ikke en bil, men hvis summen er tættere på +1, er det en bil. Dette betyder, at selvom en bil måske ikke har 4 hjul, så kan et ANN stadig kategorisere bilen korrekt, hvis bilen har mange af de andre egenskaber fundet i en bil.

2.2.2.3 Justering af weights

Justeringen af weights er det som menes, når man siger en computer "lærer". En computer kan kun "tænke" med tal, altså med det binære talsystem. Computeren justerer sine fortolkninger af vigtigheden af forskellige inputs, og dette er hvad der kaldes weight adjustments.

Måden computeren rent matematisk gør/opnår dette kan være brug af følgende formel:

$weight * input + bias$ (Ahirwar, 2017)

Bias er en variabel, som vi ikke vil beskrive dybere, end at det er upåvirket af et input.

Der findes mange forskellige måder at opskrive denne formel på, men i vores simulering, som først bliver præsenteret i et senere kapitel, har vi valgt at bruge denne formel.

For at eftervise denne formel har vi simuleret et ANN, som justerer sine weights (se bilag 2). Dette bliver der gået i dybden med i afsnit 6.

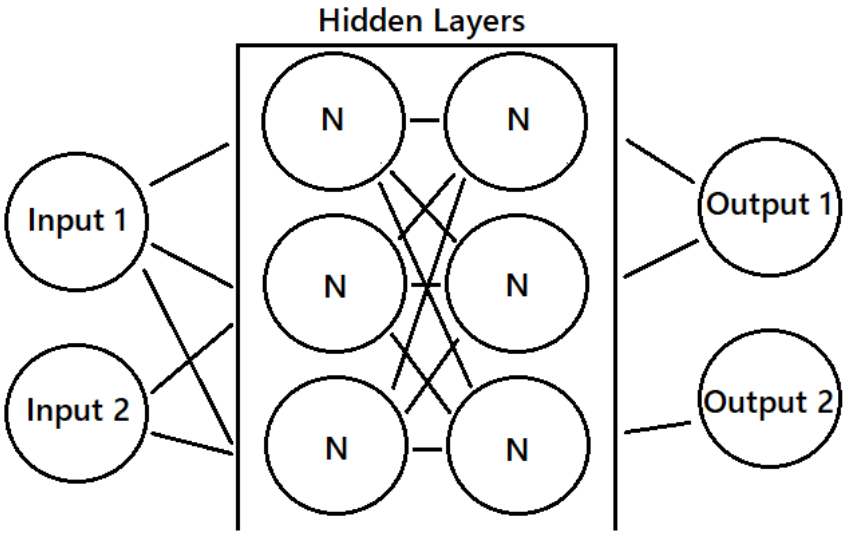
Disse justeringer af weights sker af flere omgange. Grunden til dette er, at bare fordi vægten er justeret én gang, betyder det ikke den er fejlfri, som den bestræber sig efter at være. Den skal altså lave flere gennemgange af det samme ANN, for til sidst at komme så tæt på fejlfri som muligt. Disse gentagelser kaldes epochs. Antallet af epochs er bestemt på forhånd, men et højere antal epochs er ikke universelt for mindre antal fejlvurderinger af ANN'et. Grunden til dette er, at et ANN ikke kan blive uendelig bedre. Det kan kun blive så godt, at det ikke vil have nogle fejlvurderinger. Hvis den tilfældigvis får 0 vurderingsfejl efter 3 gennemgange

eller epochs, så betyder de næste 1000 epochs ikke noget for kvaliteten af ANN'et (Mikolov, Karafiat, Burget, Cernocky & Khudanpur, 2010; s. 1046).

2.2.2.4 Hidden layers

I dette afsnit vil vi beskrive hidden layers og forklare hvorfor de er relevante, når der tales om Artificial Neural Network.

Hidden layers er grunden til, at ANN på mange måder bliver set som en black box, som betyder, at man ikke har adgang til den data, som produceres. Disse layers er hvad vi ikke kan forstå. Hidden layers bliver illustreret på det nedenstående billede.



Figur 1 - En simpel visualisering af Hidden Layers

Der skal her nævnes, at alt imellem inputtet og outputtet bliver kaldt for neuroner. Det er disse som et hidden layer består af (Maan, Jayadevi & James, 2017; s.1734 - 1744).

Man kan programmere dem til at læse dataen, som man vil have, til gengæld er dataen nærmest ulæseligt for mennesker, så man kan ikke rigtig sige præcist hvorfor den tog beslutningen, ud over at den gjorde det ud fra de parametre som man har opstillet for den.

Et Artificial Neural Network uden nogle hidden layers, ville give et input til nogle neuroner i samme layer, som så giver deres information direkte videre i form af et output. Men et ANN med flere hidden layers skal ses som flere kanaler som ANN'et skal gå igennem, før den kommer frem til et output. Dette er vigtigt, fordi, hvis formålet er, at den skal tage den rigtige

beslutning, er den nødt til at have et sted, hvor den kan vurdere dataen, som den har fået, ved brug af det som den har lært og se hvilken beslutning som er den rigtige. Det er i de hidden layers den udregner med hjælp fra en activation function. Activation function er den måde, som man har valgt den skal beregne den data, som den får. Med hjælp fra den activation function vil den se om den skal gøre noget ud fra de inputs, som den har fået, hvorefter den sender informationen videre til outputtet (Panchal, Ganatra, Kosta & Panchal, 2011; s. 332 - 337).

3. Metodeafsnit

I vores metodeafsnit vil vi dække to emner, som vi mener danner et tilstrækkeligt grundlag for at kunne komme til bunds med vores problemformulering, samt arbejdsspørgsmål.

Først vil vi introducere TRIN-modellen, som har fokus på de teknologiske artefakter og deres funktionaliteter i den selvkørende bil. Nedenstående vil TRIN-modellens seks trin blive redegjort for. Dernæst vil vi begrunde valget af at skrive en logbog sideløbende med vores rapport. Dette skyldes at det er med til at give et indblik i teoridelen "learning by doing", som er vores andet fokusområde i rapporten.

3.1 TRIN-modellen

Under dette delafsnit vil der blive redegjort for TRIN-modellen og heriblandt hvordan og hvad den bruges til, samt hvordan TRIN-modellen inspirerer til at analysere og beskrive teknologier. Akronymet TRIN står for **T**eknologi, **R**adikalt og **I**nkrementelt design i **N**etværk.

TRIN-modellen er en model, hvis formål er at beskrive en teknologi over seks spørgsmål/ trin:

1. Identikation og analyse af teknologiens **indre mekanismer og processer.**
2. Identikation og analyse af teknologiens **artefakter.**
3. Identikation og analyse af en teknologis **utilsigtede effekter.**
4. Analyse og sammenhænge i større **teknologisystemer.**

5. Opstille en **model** af teknologien.

6. Analyse, drivkræfter og barrierer for **udbredelse af innovation.**

TRIN-modellens hovedvægt ligger altså på de teknisk-videnskabelige aspekter, som findes indenfor teknologien (Jørgensen, 2018; s. 2).

3.1.1 Trin 1 Identikation og analyse af teknologiens indre mekanismer og processer

Første trin i TRIN-modellen omhandler teknologiens indre mekanismer og processer (operationelle principper), hvilket kort fortalt betyder, hvordan teknologien virker. Dette trin er inspireret af Walter Vincentis begreb om en given teknologis operationelle princip. Vincentis forklaring af begrebet 'det operationelle princip' lyder således: "in brief, how the device works", som altså betyder hvordan teknologien virker (Jørgensen, 2018; s. 29).

Dette kunne eksempelvis være en vindmølle. Så ville fremgangsmåden være at forklare dets operationelle princip. Dette kan forklares på følgende måde: Vindmøllen udnytter vinden til at få dens vinger til at rotere ved at omdanne den kinetiske energi i vinden, som driver en el generator. Ifølge de tre skabere af TRIN-modellen Thomas Budde Christensen, Erling Jelsøe og Niels Jørgensen, samt Vincenti, så antager de at alle teknologier har et operationelt princip (Jørgensen, 2018; s. 30).

Den korte forklaring af hvordan vindmøllen fungerer, i forhold til Vincentis definition af det operationelle princip, er altså hvordan teknologien (i dette tilfælde vindmøllen) fungerer:

Det kan sagtens lade sig gøre at redegøre for en teknologis operationelle princip og ikke redegøre for komponenternes operationelle principper. I forhold til vindmøllen, så tjener vingerne i sig selv ikke et formål for at løse et problem for menneskene, men den har et formål som er en del af en større enhed, som er vindmøllen. Men man kan kigge på de allermindste artefakter som er del af teknologien, og analysere på hvordan de er med til at skabe den teknologi som bliver analyseret på.

3.1.2 Trin 2 Identikation og analyse af teknologiens artefakter

Trin to handler om at belyse hvilke artefakter, som indgår i en given teknologi, og så at beskrive artefakterne. Et artefakt er en genstand, skabt af mennesker. Når der tales om

artefakter i TRIN-modellen, menes der tekniske artefakter. Tekniske artefakter skal forstås som, at de har en praktisk eller formål i forhold til teknologien. Denne formålsbestemthed er til forskel fra artefakter i kunst, eksempelvis et maleri (Jørgensen, 2018; s. 30).

Hvis vi fortsat kigger på vindmøllen, kan vi se hvilke af vindmølleteknologiens elementer, der indgår. Eksempelvis nogle af de tekniske artefakter som vindmøllen består af, en generator eller bremsen, som findes inde i vindmøllehuset. Som nævnt tidligere, så bliver generatoren drevet af vingerne, som bliver drevet af vinden. Bremsen er blevet tilføjet til vindmøllen, så vindmøllen ikke lider overlast, hvis det skulle blæse for meget. Intentionen af det tekniske artefakt og den medvirkende funktion hænger sammen med de indre mekanismer og processer for at skabe den ønskede funktionalitet.

Når man taler om vindmølleteknologiens elementer og artefakter, er det også være relevant at nævne den viden, som ligger bag artefakterne. Vindmøllen er designet, så den har gode aerodynamiske vinger. Vingerne udnytter ikke kun vindens tryk, men også luftstrømmen omkring vingerne, Luftstrømmen skaber et undertryk på vingens bagside i forhold til vinden (Danmarks vindmølleforening, 2013). Det kan også være værd at nævne andre egenskaber ved artefakterne, såsom størrelser og længder på eksempelvis vingen, eller hvilket materiale det består af.

3.1.3 Trin 3 Identikation og analyse af en teknologis utilsigtede effekter

Hovedformålet med trin tre, identifikation og analyse af teknologiens utilsigtede effekter, er fokus på de uønskede effekter (Jørgensen, 2018; s. 41 - 44).

Hvis en vindmølle betragtes, ville vindmøllens tilsigtede effekt, altså dens funktion, være at generere strøm. Men i trin tre analyseres der på de utilsigtede effekter, som vurderes at være negative. De utilsigtede effekter kan opstå i flere former. Eksempelvis kan nogle af disse utilsigtede effekter kategoriseres som vedvarende effekter, som når vindmøllen støjer eller ødelægger udsigten. Det er ikke noget som bare stopper eller opstår på bestemte tidspunkter eller under bestemte forhold. Men ikke alle disse effekter er vedvarende. Nogle effekter er kun tilstedeværende under uheldige situationer og er derfor kategoriseret som risici. Disse effekter kan opstå af mange forskellige årsager. Vindmøllen kan gå i stykker, hvilket kan forårsages af misvedligeholdelse eller eventuelt dårligt design (Jørgensen, 2018; s. 41 - 44).

I nogle tilfælde kan man løse de vedvarende utilsigtede effekter, ved at opfinde ny- eller forbedre teknologien. I forhold til vindmøllen, kunne det være at man producerede en generator, som ikke danner den samme mængde støj som den nuværende.

3.1.4 Trin 4 Analyse og sammenhænge i større teknologisystemer

Det fjerde trin i TRIN-modellen beskæftiger sig med de større teknologi-sammenhænge, som en teknologi er en del af. Her betragtes der en given teknologi, som indgår i større sammenhæng i form af en enhed eller et system, og hvordan disse forskellige teknologier, artefakter, elementer eller eventuelle aktører, skaber det samlede teknologiske system, som ses på.

Disse teknologisystemer besidder en samlet bestemt funktionalitet, som tillader en ændring af natur med tanke på at opfylde et menneskeligt behov.

En af grundene til at det er vigtigt at analysere sammenhængene i det samlede teknologiske system er, at væsentlige utilsigtede effekter ved teknologien kan hænge sammen med andre elementer i det teknologiske system. Disse utilsigtede effekter i det teknologiske system er måske ikke noget man i første omgang har tænkt på som det centrale (Jørgensen, 2018; s. 44 - 47).

Eksempelvis er vindmøllen et eksempel på et teknologisk system. Eftersom vindmøllen består af en masse teknologier med forskellige funktionalitet, der tilsammen er med til at skabe det teknologiske system, som vindmøllen kendetegnes ved.

”*Møllens vindfane retter sig ind efter vinden, nøjagtig som en vejrhane. I enden af vindfanens lodrette aksel er en sensor, som aktiveres, når vindfanen ændrer retning. Sensoren giver signal til at starte krøjemotoren, som drejer hele kabinen, så rotoren vender op imod vinden. Vinden blæser nu omtrent vinkelret ind på rotoren*.”(Danmarks vindmølleforening, 2013; s.1).

Møllens vindfane, sensor og krøjemotor er teknologiske artefakter, som alle er en del af vindmøllens teknologiske system og er med til at bidrage til vindmøllens funktionalitet. Vindmøllen består af mange flere teknologier, artefakter og aktører, der hver især har deres egne delfunktioner, som er med til at drive vindmøllen. Hvis de tre teknologier i eksemplet ikke var inkorporerede i vindmøllen, ville den ikke være i stand til at udnytte vinden til det

fuldeste, samtidig med at vindmøllen for det meste ikke ville være rettet mod vinden, og derfor ikke kan udnytte den.

Man kan derfor sætte sit fokus på en del af det teknologiske system eller systemer, som man betragter. Ved brug af trin fire i en teknologisk analyse anbefales det, at man giver en begrundelse for de elementer, som man sætter fokus på i det teknologiske system. Begrundelsen skal naturligvis henvises til analysens fokus (Jørgensen, 2018; s. 44 - 47).

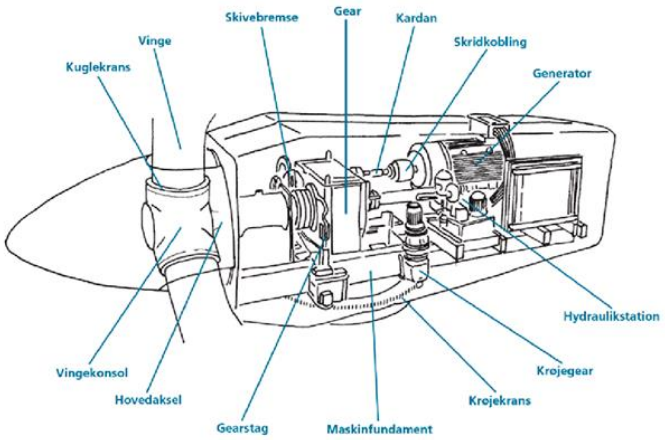
3.1.5 Trin 5 Opstille en model af teknologien

I dette trin er der fokus på forskellige typer af tekniske modeller af teknologi, evt. beregningsorienterede eller abstraktioner af modeller. Disse typer af modeller kan blandt andet bruges i mere konkrete designbeslutninger, eksempelvis visualisering af et vindmøllehus (Jørgensen, 2018; s. 47 - 49).

Men en model er også noget, som kan være abstrakt, visuelt eller en fysisk repræsentation af et fænomen eller en genstand. Specifikt udvalgte egenskaber ved fænomenet eller genstanden søges afprægede og- eller undersøgt. Endvidere kan en model også være et epistemisk værktøj til at skabe eller udvikle konkrete artefakter. Det skal lige nævnes, at alle modeller er abstraktioner, eftersom de forsimpler og fremhæver særlige aspekter ved et design eller fænomen.

Illustrationer kan også være en type af modeller. Denne form for visuel model anvendes til at repræsentere en ting eller et fænomen. Visuelle modeller kan benyttes til at illustrere, forklare eller undersøge hvordan ting ser ud, hvordan de fungerer, som i form af de operationelle principper eller hvordan ting eventuelt kan udvikle sig.

Et eksempel på denne type model kunne være af vindmøllehuset. Her illustreres det hvorledes vindmøllen opererer, og hvordan den er konstrueret.



(Danmarks vindmølleforening, 2013; s. 3)

3.1.6 Trin 6 Analyse, drivkræfter og barrierer for udbredelse af innovation

Det sjette og sidste trin i TRIN-modellen handler om udbredelsen af en given innovation og hvilke drivkræfter der er med til at udbrede denne. Dette trin kunne blandt andet inddrage Everetts diffusionsteori, omkring udbredelsen af en innovation.

Everett M. Rogers beskriver diffusion som "*the process in which an innovation is communicated through certain channels over time among members of a social system. "(*Rogers, 2003; s. 5). Han ser diffusionen af en innovation som en proces, hvor innovationen bliver spredt gennem kommunikation. Everett har også opstillet nogle "characteristics of innovations", som er fem punkter han finder relevant i forbindelse med at overtale en potentiel bruger til at adoptere en innovation. Disse punkter er kort beskrevet:

1. **Relative advantage**

Den relative fordel er fordelen for brugeren ved at bruge en innovation.

Denne fordel kan handle om det økonomiske og sociale aspekt, såvel som følelsen af tilfredsstillelse eller belejlighed af innovationen (Rogers, 2003; s. 15).

2. **Combability**

Hvis en innovation skal kunne implementeres, er det en stor fordel hvis innovationen er kompatibel med det nuværende samfund. Rogers argumenterer for, at innovationen skal være

"consistent with the existing values, past experiences and needs of potential adapters" (Rogers, 2003; s. 15).

Udbredelsen af en innovation vil foregå hurtigere, hvis den passer ind i det eksisterende samfund. Eksempelvis kunne telefoner med touchscreen bruges med de gamle telefoner, hvorimod en helt ny form for kommunikation, som ikke kunne bruges sammen med de nuværende telefoner, formentlig ikke ville udbredes nær så hurtigt.

3. **Complexity**

Kompleksiteten af en innovation har ifølge Rogers også stor indflydelse på udbredelsen af en idé. Hvis en innovation er svær at bruge eller forstå, vil det tage længere tid at få flere til at bruge det. Simple innovationer bliver ofte hurtigere adopteret (Rogers, 2003; s. 16).

4. **Triability**

Dette punkt handler om muligheden for at eksperimentere og afprøve en idé. Hvis en potentiel bruger kan afprøve et produkt, er der større sandsynlighed for at de vil adoptere det.

5. **Observability**

En synlig innovation kan starte samtaler og diskussioner. Dertil vil naboerne ofte spørge ind til erfaringer og en evaluering med denne innovation, og på den måde kan synligheden af innovationen og synligheden af innovationens resultater have betydning for diffusionen (Rogers, 2003; s. 16).

Udover Everetts diffusionsmodel, kunne man også inddrage barriereanalyse, hvor man analyserer på hvilke forhindringer og udfordringer der står i vejen for udbredelsen af en innovation.

3.2 Logbog

Sideløbende med konstruktionen af simulationen, har vi valgt at skrive en logbog. En logbog minder på nogle punkter om en dagbog. I logbogen vil vi skrive de tanker, idéer og forventninger ned, som vi støder på under konstruktionen af simulationen. Det er et godt redskab til at øge læringen, både fordi man får sat ord på ens tanker, samt man altid kan gå tilbage og viderebygge eller anvende de overvejelser og tanker man tidligere har gjort sig i

processen, hvis man eksempelvis støder på en skriveblokade eller bare har brug for inspiration (Social- og Sundhedsskolen Viborg Amt, s.d)

Begrundelsen for at inddrage logbogen er, at det hænger rigtig godt sammen med '*learning by doing'* teorien. Både teorien og logbogen er en proces over tid, hvor de komplimenterer hinanden godt. I vores tilfælde vil logbogen også blive inddraget som belæg for den læring vi har opnået gennem simulationen, og generelt til at beskrive selve læringsforløbet, eksempelvis om resultatet blev det samme, som vi først havde forventet eller om blev det noget helt modsat. Selve logbogen er indskrevet som bilag, og vil blive refereret til i analysen.

Opsamling

Vi har i dette kapitel redegjort for TRIN-modellen, og en begrundelse for at inddrage en logbog, i vores rapport. I TRIN-modellen blev der beskrevet hvordan, de seks trin kunne give en forståelse af en given teknologi. Disse trin beskrives i følgende rækkefølge: 1. Hvordan en teknologis teknologiske artefakter er med til at give den overordnede teknologi et formål (operationelle princip). 2. Hvilke typer for teknologiske artefakter som vi har valgt at sætte fokus på, samt hvilke egenskaber de har. 3. Om der findes eventuelle utilsigtede effekter ved den givne teknologi, som betragtes. 4. Det overordnede teknologiske system, altså hvordan teknologierne tilsammen er med til at skabe den givne teknologi. 5. Visualiseringer af forskellige type modeller som bruges til at beskrive mere konkrete designbeslutninger. 6. Udbredelse af en innovation og hvilke drivkræfter og barriere der er med til at udbrede en given innovation.

Til sidst har vi valgt at inddrage en begrundelse for at bruge en logbog. Logbogen giver mulighed for at nedskrive sine erfaringer, overvejelser og forventninger ned, og arbejder godt sammen med "learning by doing", og på den måde er med til at øge indlæring, eftersom det er lettere at reflektere over sine handlinger.

4. Kort om selvkørende biler

I dette afsnit vil vi nævne hvad der menes med en "selvkørende bil" og dertil gennemgå de forskellige niveauer, som klassificerer hvor autonom en bil er, samt gennemgå nogle udvalgte projekter og pejlemærker, som har haft en direkte indflydelse på udviklingen af selvkørende biler.

4.1 Niveauer af autonomi

En selvkørende bil er en "autonom" bil. En autonom, selvkørende eller førerløs bil henviser til en bil, som kører uden eller med meget lidt menneskelig kontrol og indvirken. Society of Automotive Engineers (SAE) har udarbejdet en tabel/model med seks forskellige niveauer/levels af autonomi hos biler, som går fra ingen autonomi til en fuldt autonomisk bil. Denne model vil blive gennemgået level for level nedenfor.

Level 0 - Ingen autonom

Bilen har ingen former for autonomi og det er 100 procent føreren, som styrer bilen og vælger hvad der skal ske (Hecht, 2018).

Level 1 - Intelligente egenskaber

Bilen har nogle intelligente egenskaber (Hecht, 2018). Den er udstyret med driver assistent, som kan hjælpe chaufføren og komme med advarsler. Driver assistent overtager ikke styringen (Watzenig & Horn, 2017; s. 4).

Level 2 - Lidt autonom

Bilen har en form for autonomi. Her kan driver assistent hjælpe med at styre bilen over kort tid, hvis der eksempelvis kommer en form for advarsel. Det kan eksempelvis være hvis føreren falder i søvn, og trækker over i den anden vejbane. Her kan den dreje rattet, så bilen kommer over i den rigtige vejbane igen. Den kan også hjælpe med at accelerere eller decelerere. Føreren er stadig den, som har styringen over bilen (Watzenig & Horn, 2017; s. 4)

Level 3 - Nominal autonom

Bilen begynder at kunne køre over længere strækninger, uden chaufføren har hænderne på rattet, men stadig har øjnene på vejen konstant. Chaufføren skal dog kunne overtage styringen med det samme (Hecht, 2018).

Level 4 - Høj autonom

Her begynder bilen rigtig at være autonom. Bilen begynder at have diverse sensorer; LiDAR, radar, ultralyd og kameraer mm., som hjælper med at detektere omgivelserne. Der er stadig

rat og pedaler, så chaufføren kan tage over når det behøves. Det kan være hvis bilen kommer ud for et scenarie, som den ikke kender. Chaufføren behøver ikke være opmærksom på vejen hele tiden og kan eksempelvis læse en bog eller kigge på telefonen. Bilen styrer, accelerer, decelererer, drejer selv osv. (Hecht, 2018).

Level 5 - Fuldt ud autonom

Bilen har stadig en masse sensorer, som detekterer omgivelserne. Den er nu selvkørende og kan køre under alle omstændigheder, uden nogen interaktion fra mennesker. Da bilen ikke har brug for en chauffør, indeholder den længere ikke et rat eller pedaler. Passagerne kan derfor eksempelvis sove imens de kører fra A til B (Hecht, 2018).

4.2 Historien om udviklingen af selvkørende biler

I dette afsnit vil vi kort nævne nogle relevante nedslag i historien, som har haft indflydelse på udviklingen af selvkørende biler.

Linriccan/American Wonder 1926

" The timeline of autonomous cars begins in 1926 with world's first radio controlled car- 'Linriccan Wonder'." (Bimbraw, 2015; s. 191)

Virksomheden Houdina Radio Control stod bag "American Wonder" (også kaldet 'Linriccan Wonder'), som var det første store skridt mod selvkørende biler. Bilen var radiokontrolleret ved hjælp af en antenne på taget og blev styret fra en anden bil, som kørte lige bagved (Bimbraw, 2015; s. 191).

Guided by wires 1953 - 1958

I 1953 byggede RCA Labs en miniature bil, som blev styret af ledninger, som blev lagt i gulvet. Dette ledte senere til at Leland Hancock og L. N. Ress blev inspirerede og eksperimenterede med systemet på en motorvej i 1958 ved hjælp af et elektrisk kredsløb, som blev lagt i asfalten over 121,92 meter (Bimbraw, 2015; s. 192).

Citroën DS

I 1960'erne testede TRL (United Kingdom's Transport and Roads Research Laboratory) deres selvkørende bil, Citroën DS, som gjorde brug af magnetiske kabler, som var indlejret i vejen (Bimbraw, 2015; s. 193).

Mercedes-Benz robotic van

I 1980'erne begyndte flere og flere selskaber og firmaer, at forske og eksperimentere med selvkørende biler. En af flere projekter med selvkørende biler var Mercedes-Benz robotic van. Den var "vision-guided", hvilket betød bilen gjorde brug af sensorer og kameraer til at styre og køre, i stedet for kabler som tidligere. Bilen nåede op på 63 km/t på veje uden trafik (Bimbraw, 2015; s. 193).

DARPA (ALV)

DARPA (Defense Advanced Research Projects Agency) havde også stor indflydelse på udvikling af ny teknologi og generelle fremskridt af selvkørende biler. DARPA havde et projekt kaldet ALV (Autonomous Land Vehicle), som fokuserede på at udvikle en selvkørende bil, som kunne være til gavn for det amerikanske militær. Dette indebar blandt andet en selvkørende bil som kunne holde til det hårde terræn (Bimbraw, 2015; s. 193). Projektet var også blandt de første til at demonstrere brugen af LiDAR-sensorer i en selvkørende bil.

> *"The ALV project achieved the first road- following demonstration that used computer vision, LiDAR and autonomous control to direct a robotic vehicle at speeds of up to 31 km/h (Davis, 1987; Leighty, 1986; Lowrie, 1985; Chandran, 1987). "* (Bimbraw, 2015; s. 193)

Waymo (Google)

Waymo startede som Google Self-Driving Car Project i 2009, men i 2016 blev til det til en separat division, som blev kaldt 'Waymo'. Waymo er en virksomhed, som arbejder med at udvikle og forbedre deres selvkørende biler. De har allerede nu (anno 2019) færdigudviklet en selvkørende bil, testkørt den på offentlige veje (med og uden rat og chauffør), samt lanceret en app, som brugeren kan benytte til gøre brug af deres taxaservice 'Waymo-one', som bruger selvkørende biler (Waymo LCC, 2018).

Waymo-one er ikke den første taxaservice til at gøre brug af selvkørende biler, men er den virksomhed som er nået længst i udviklingen. Blandt andet har Uber også forsøgt sig med

selvkørende taxaer, samt pilotprojekter i Dubai, hvor der dog stadig er en chauffør bag rattet for en sikkerheds skyld (Lekach, 2018).

4.3 Opsummering

I dette afsnit har vi gennemgået nogle grundlæggende aspekter af selvkørende biler. Når man snakker om en autonom, selvkørende eller førerløs bil, bliver der som oftest refereret til den samme tankegang om en bil, der selv kan køre. Society of Automotive Engineers (SAE) har udviklet en tabel over 6 forskellige niveauer er autonomi, når det kommer til netop selvkørende biler. Disse niveauer går fra 0 til 5, hvor 0 er bil uden nogen form for autonomi og 5 er en 100% selvkørende bil, uden menneskelig indgriben.

Dertil har vi givet et kort historisk overblik over udviklingen af idéen om en selvkørende bil. Et af de første vigtige skridt blev taget i 1926 med Houdinas radiokontrollerede bil. Sidenhen har der været en lang række eksperimenter og projekter, som har været med til at udvikle teknologier og idéer til at lave en selvkørende bil. Et af de firmaer, som er nået længst mod målet med en fungerende selvkørende bil, er Waymo. De har allerede noget, der minder om en færdig selvkørende bil på level 5, og har allerede lanceret en taxiservice, hvor de gør brug af disse biler.

5. Hvordan fungerer de teknologiske artefakter i en selvkørende bil og hvilken funktionalitet har de i det teknologiske system?

I dette afsnit vil der blive gået i dybden med analyser af hvordan de teknologiske artefakter i den selvkørende bil fungerer, samt hvilken funktionalitet de har i det teknologiske system.

Der vil først blive udarbejdet en analyse af den selvkørende bils indre mekanismer og processer, på baggrund af det operationelle princip, som skal belyse de teknologiske artefakters funktionalitet og formål, ud fra TRIN-modellens første trin.

Dernæst vil der blive fremvist en identikation af diverse artefakter, som indgår i den selvkørende bil, som vi finder relevante for emnet, samt en analyse ud fra TRIN-modellens andet trin.

Og til sidst vil der blive givet en analyse af de teknologier i sammenhæng med det større teknologiske system der findes i den selvkørende bil. Der vil her blive analyseret på hvordan de forskellige teknologiske artefakter arbejder på kryds og tværs af hinanden for at skabe den ønskede teknologiske funktionalitet, ud fra TRIN-modellens fjerde trin.

5.1 Analyse af teknologiens indre mekanismer og processer

Der vil i dette afsnit blive givet en indentikation og analyse af teknologien "den selvkørende bils" indre mekanismer og processer, altså det operationelle princip.

Den selvkørende bils operationelle princip er at kunne transportere mennesker fra A til B på en sikker måde, uden indgriben fra menneskets side af. Måden den gør dette på er gennem flere former for teknologi, der tilsammen arbejder om at skabe en kunstig intelligens, som giver bilen mulighed for at handle på egen hånd. Den selvkørende bil består af en GPS, og en række kameraer og sensorer. Ved brug af flere forskellige sensorer og kameraer, vil der ske en "sensor fusion". En sensor fusion er hvor bilens styresystem kombinerer og analyserer sensorernes inputs. Herved træffer bilen en masse beslutninger. Det kan være om den eksempelvis skal køre eller bremse (Hecht, 2018).

Disse teknologier videregiver den information de opfanger via forskellige former for bølger, til et AI-styret program, i form af Machine Learning. På denne måde, kan den selvkørende bil selv bestemme dens næste handlinger udefra, og dermed efterkomme menneskets behov om en sikker transportmulighed.

5.2 Analyse af teknologiens artefakter

Dette afsnit vil fokusere på trin to i TRIN-modellen, som omhandler de teknologiske artefakter, som indgår i en selvkørende bil. Der vil være fokus på de teknologiske artefakter, som skiller sig ud fra en almindelig personbil, og som gør bilen i stand til at være selvkørende.

5.2.1 Sensorer

De første relevante artefakter, i en selvkørende bil, er bilens forskellige sensorer.

5.2.1.1 LiDAR

LiDAR står for light detection and ranging, I nogle tilfælde bliver LiDAR også refereret som LaDAR, som står for laser detection and ranging. LiDAR og LaDAR er det samme. På det grundlag har vi valgt at skrive LiDAR i rapporten (Khader & Cherian, 2018; s. 2). LiDAR-sensoren har til formål at lokalisere bilens omgivelser. Det er altså bilens måde at se på. LiDAR er en teknologi, som bruges til at indsamle store mængder af afstandsmålinger, ved hjælp af laserstråler. Alt efter hvor lang tid der går fra laseren bliver sendt afsted til den kommer tilbage, kan sensoren beregne distancen til det objekt laseren stødte på (Khader & Cherian, 2018; s. 2). LiDAR bruger ikke kun det synlige spektrum, men også ultraviolet og infrarød stråling (Fujii & Fukuchi, 2005; s. 3). Med denne data bliver der genereret et virtuelt 3D-kort, som består af en masse prikker. Disse prikker symboliserer, der hvor laseren stødte på noget, og det danner deraf et virtuelt billede af omgivelserne, konstrueret af prikker. Dataen giver softwaren mulighed for at lokalisere placeringen af mennesker og objekter, som er rundt om bilen (Hecht, 2018). LiDAR kan detektere genstande i en distance fra få meter til mere end 200 meter. Dog har LiDAR svært ved at detektere genstande, som er helt tæt på den selvkørende bil, og derudover har LiDAR sværere ved at orientere sig, når sigtbarheden er dårlig. Det kan eksempelvis være hvis det regner, sner eller det er tåget. Til gengæld kan LiDAR stadig navigere rundt, når det er mørkt (Khader & Cherian, 2018; s. 3). Da en LiDAR kun kan måle afstande, så kan LiDAR ikke læse hvad der står på færdselstavler og andre tavler, da de har en flad overflade (Khader & Cherian, 2018; s. 2)

Der findes to typer af LiDAR-sensorer. Mechanical LiDAR og solid-state LiDAR. I forhold til de andre sensorer, så har en Mechanical LiDAR en roterende enhed oven på bilen, som sørger for at der er dækket 360 grader rundt om bilen, og herved har det bredeste synsfelt. Ved brug af Solid-state LiDAR, så sidder sensorerne på hver side af bilen, i stedet for ovenpå bilen (Khader & Cherian, 2018; s. 4).

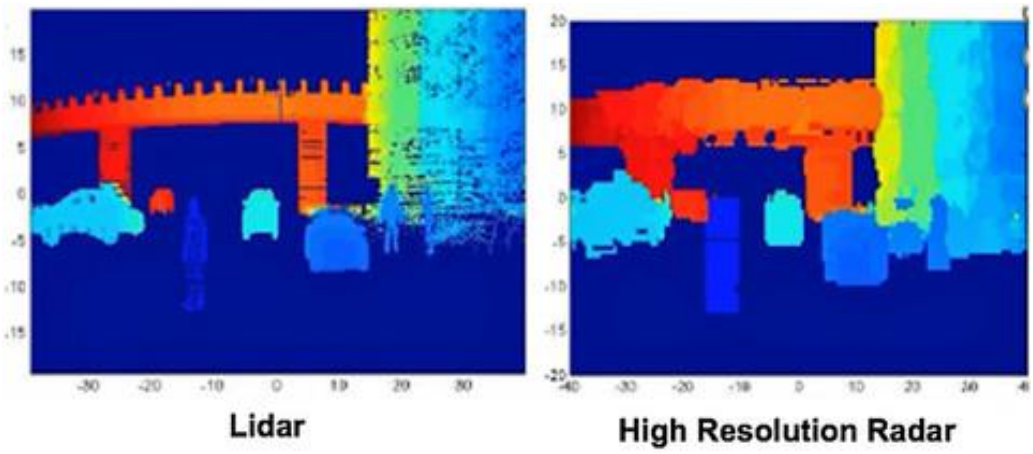
5.2.1.2 Radar

En anden sensor, som er værd at nævne i den selvkørende bil, er radar. Radar står for Radiowave Detection And Ranging. Den fungerer ved hjælp af radiobølger (Fujii & Fukuchi,

2005; s. 3). Radar-sensorens formål er at detektere genstande, som enten kommer forfra, bagfra eller fra siden. Ligesom LiDAR-sensoren, så afhænger dataen af hvor lang tid der går, inden signalet vender tilbage (Hecht, 2018). I modsætning til LiDAR-sensoren, så er radar-sensoren mere stabil i dårligt vejr, eksempelvis når det sner, regner eller er tåget (Khader & Cherian, 2018; s. 3).

Der findes både radarer, som virker på kort og lang afstand. Radar-sensoren kan detektere genstande fra under en meter til mere end 200 meter. (Khader & Cherian, 2018; s. 3). Radar-sensoren som virker på kort afstand, er 24 GHz short/medium-range radaren, som bliver brugt til at detektere de blinde vinkler på bilen. Den er behjælpelig i forbindelse med vognbaneskift og giver advarsler om kollisioner bagfra (Hecht, 2018).

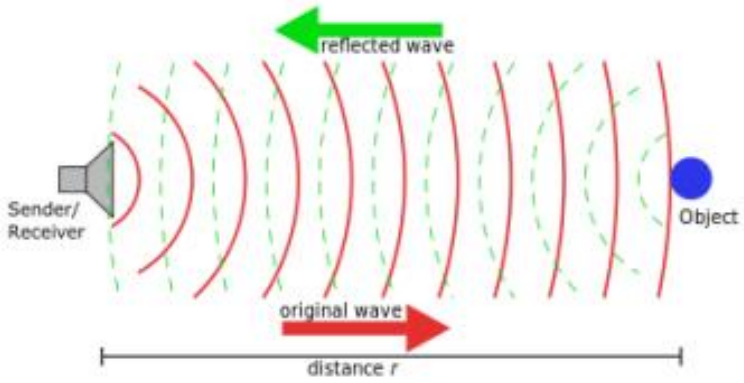
77 GHz Long Range radar-sensoren kan ligesom LiDAR-sensoren detektere mennesker og objekter, som er foran bilen i op til 200 meters afstand, dog har den svært ved at se små ting jo længere væk den skal detektere, til modsætning af LiDAR (Khader & Cherian, 2018; s. 3). Dette er repræsenteret på nedenstående billede.



(Neal, 2018)

5.2.1.3 Ultralyd

Ultralyd, som også er kendt som Sonar, står for Sound Navigation and Ranging (Fujii & Fukuchi, 2005; s. 3). Ultralydssensoren bruger ekkolokation til at navigere med. Ultralydssensoren udsender lydbølger for at detektere sine omgivelser, som også kan ses på billedet nedenfor (Lim, Keoh & Thing, 2018; s. 3).



(Lim, Keoh & Thing, 2018; s. 3)

Ultralydssensoren har ikke en særlig stor rækkevidde, derfor er den god til at detektere objekter, som er relativt tæt på. Ultralydssensoren bliver brugt i parkeringsassistenter.

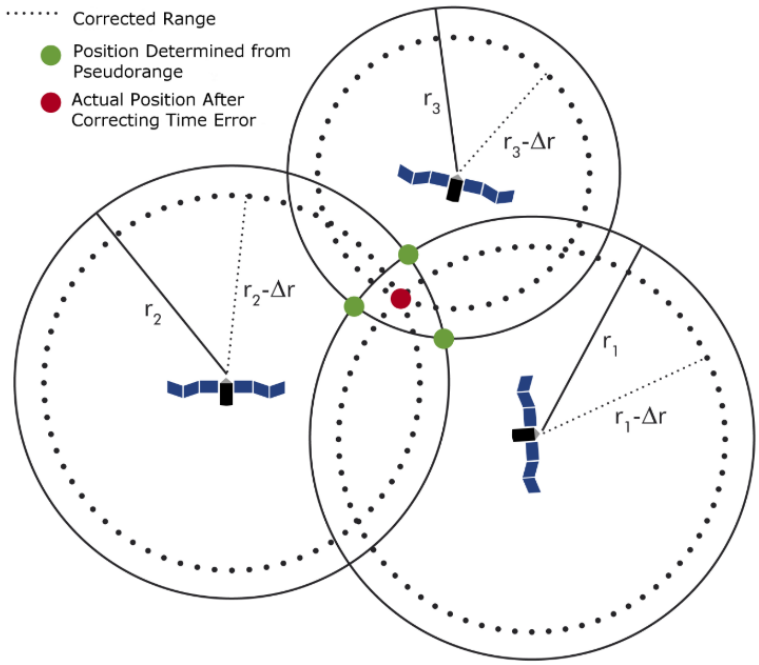
En parkeringsassistent er en egenskab, som hjælper med at parkere bilen. Ultralydssensoren er med til at detektere objekterne, så bilen ved hvor meget hjulene skal dreje imens den parkerer (Lim, Keoh & Thing, 2018; s. 2).

5.2.2 GPS

I de selvkørende biler vil der være en GPS, som har til formål at finde vej til destinationen. GPS'en vil formentlig ikke guide selve bilens eksakte placering på vejen, eksempelvis i forhold til andre bilister eller fodgængere, men ville navigere bilen til hvilken retning den skal køre.

GPS står for Global Positioning System og virker via satellitter i rummet, som sender et signal til jorden med satellittens position i rummet. På jorden beregnes afstanden til satellitten ved at udregne $distance = hastighed * tid$(Olson, 2018).

Måden hvorpå satellitterne kan bestemme den specifikke position er ved brug af trilateration. Som det er visualiseret på billedet, bruges mindst tre kendte positioner (fra satellitter) til at bestemme positionen af, i vores tilfælde, bilen (Olson, 2018). Dette er illustreret nedenunder.



A two-dimensional representation of GPS trilateration. A GPS receiver determines its position by calculating "pseudoranges" r1, r2, and r3 and adjusting for timing inaccuracies to correct range error, Δr. In the real three-dimensional world, ranges to four satellites are needed to determine position. Source: SEOS Project

(Olson, 2018)

5.2.3 Kamera

Den sidste teknologi, som vi har valgt at fokusere på, er kameraerne. Kameraerne er den eneste teknologi i den selvkørende bil, som kan se farver og er stærkt afhængig af de rigtige lysforhold, for at kunne virke optimalt. Kameraerne kan derfor ikke se i mørke, som LiDAR-sensoren kan. Modsat sensorerne, så kan kameraet ikke måle afstande (Khader & Cherian, 2018; s. 3).

Kameraerne i den selvkørende bil bruges eksempelvis til at læse vejskilte og lysreguleringer. Herved ved den om den har tilladelse til at køre, hvor hurtigt den må køre og om der eksempelvis er vejarbejde tæt på (Hecht, 2018).

5.2.4 Artificial Intelligence

Artificial Intelligence er hjernen i den selvkørende bil og bliver brugt til at indsamle og bearbejde alle informationerne i de selvkørende biler. AI, samt Artificial Neural Network og Machine Learning, blev uddybet i teoriafsnittet 2.2.

5.3 Sammenhænge i større teknologisystemer

I dette afsnit vil vi analysere på sammenhængene i det teknologiske system i en selvkørende bil. Vi har valgt at analysere på de specifikke dele af den selvkørende bils teknologiske artefakter, som vi mener er det som er med til at skabe den selvkørende bil og dermed skille sig ud fra en almindelig fører dreven bil. Der findes mange flere teknologier og artefakter i den selvkørende bil som vi ikke vælger at berører, heriblandt dens motor, eftersom det ikke er den i sig selv som tillader den selvkørende bil at handle på egen hånd. På baggrund af dette lægger vi vores fokus på de teknologiske artefakter, som vi mener, er med til at skabe den enhed vi undersøger og analyserer på. Denne enhed består af et større sammenhængende af teknologier som vi har kategoriseret til at være:

- Artificial Intelligence, herunder Artificial Neural Network og Machine Learning
- LiDAR
- Radar
- Ultralyd
- GPS
- Kamera

Ud fra de ovenstående teknologiske artefakter mener vi, at hver af disse teknologiske artefakters delfunktioner i systemet tillader at ændre bilens natur fra at være et transportmiddel, som er afhængig af en fører for at kunne navigere og køre forsvarligt, til at være en selvkørende bil, der kan handle hurtigere og bedre end mennesker uden menneskelig indgriben.

Alle disse teknologiske artefakter er som sagt en del af den selvkørende bils system. Artefakterne arbejder på kryds og tværs af hinanden for at skabe dette teknologiske system. Hver især gør de ikke meget for at skabe den enhed vi undersøger, men sammen, i det større teknologi-sammenhæng, tillader de at skabe en selvkørende bil. Vi vil her tage udgangspunkt i de tre punkter på illustrationen i bilag 1.

Bilens AI arbejder ud fra Machine Learningens ANN, som fungerer som bilens data center/hjerne, som ses i punkt 2 (bilag 1). Her bearbejder den selvkørende bil alt den data den

modtager fra sine sensorer, som er illustreret i punkt 1 (bilag 1). Når bilens LiDAR, Radarer, ultralyd eller kamera opfanger noget eller modtager nye data, bliver disse data videresendt til bilens AI, hvorefter AI'en vil reagere på de data den modtager. På samme måde fungerer GPS'en. Den modtager data omkring den selvkørende bils position, og kender til den destination, som er det ønskede mål at komme frem til.

Som sagt gør de teknologiske artefakter ikke meget hver for sig. De bygger på hinandens styrker, og dækker for hinandens svagheder. Eksempelvis bruges de forskellige sensorer i forskellige situationer. Nogle sensorer virker bedre på nogle afstande end andre, og eksempelvis er LiDAR-sensoren ikke i stand til at aflæse vejskilte, derfor er der brug for andre teknologier, som kan tage over, hvor der fejles.

En utilsigtet effekt i forhold til selvkørende bilers sammenhængende system kan skyldes GPS'en. Det er nødvendigt, for at den virker, at den har en forbindelse til de satellitter der transmitterer data, for at GPS'en kan modtage data, som der så kan videregives til AI'en. Hvis GPS'en ikke har en forbindelse, er det ikke længere muligt for bilen at finde vej til sin destination. Denne utilsigtede er ikke vedvarende og det er ikke sikkert at GPS skaber problemer. Derfor er dette kategoriseret som risici.

En sidste nævneværdig utilsigtet effekt ved den selvkørende bils teknologisystem, stammer fra det Artificial Neural Network bilen bruger. Eftersom denne AI er baseret på Machine Learning, lærer computeren på egen hånd. Dette medfører at man ikke ved præcist hvordan at AI'en har lært de ting som den nu lærer. Dette er ikke til at afkode, og derfor opstår der en form for black box koncept omkring AI'en. Hvis der opstår en fejl med den selvkørende bils styresystem, er det altså ikke til at vide hvor problemet opstod eller hvordan det umiddelbart kan løses.

5.4 Delkonklusion

Det operationelle princip i en selvkørende bil er kort sagt at transportere mennesker. Dertil kommer det, at det skal ske uden menneskelig indgriben og på en sikker måde. Den selvkørende bil opnår dette ved brug af mange teknologier. Vi har valgt at fokusere på nogle af de teknologier, som har direkte indvirkning på at bilen kan være selvkørende. Dette er blandt andet en række sensorer, herunder LiDAR-sensoren, som kan tegne et virtuelt 3D-kort af omgivelserne ud fra lasermålinger, samt både radar og ultralyds sensorer. De beskrevne

teknologier arbejder sammen i et teknologisk system, hvor teknologier arbejder sammen for at den selvkørende bil kan opfylde sit operationelle princip.

6. Eksperiment med justering af weights

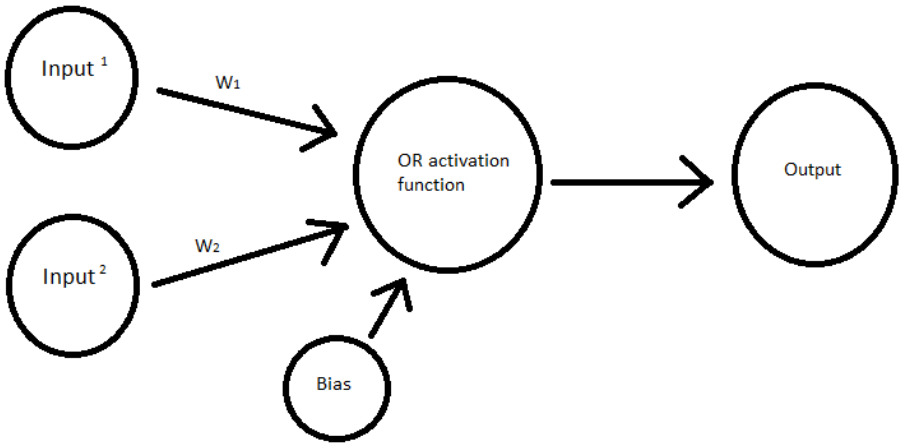
I de følgende afsnit af dette kapitel, vil det blive gennemgået, hvordan simulationen af weight adjustments fungerer, og hvordan computeren i form af et Artificial Neural Network lærer på, samt hvordan den tilpasser sig nye situationer, ved at ændre på de weights den modtager.

Dette er relevant i forhold til, at vores produkt ikke vil have fokus på selve ANN'et. Men for at forstå hvordan ANN'et kunne fungere, så har vi lavet en enkeltstående simulation af hvordan ANN'et kan justere sine weights. Der skal her nævnes, at der er mange forskellige teorier og formler på dette emne, så dette er kun én måde at justere weights på, men der er andre muligheder. ANN, herunder weights, foregår i punkt 2 i bilag 1.

6.1 Simulation af weight adjustments

Gennemførelsen af denne simulation af et Artificial Neural Network, som justerer sine weights i forhold til inputtet og det korrekte output, har til formål at give os indsigt i hvordan et ANN i realiteten justerer sine weights.

Inden vi startede konstruktionen af denne simulation, havde vi en hypotese om, at weights bliver justeret i forhold til hvad outputtet kræver. Med dette mener vi, hvis det korrekte output kræver en weight på eksempelvis 0.3 så ville weights blive justeret efter dette.
For at få en bedre forståelse af weight adjustments i et ANN har vi besluttet at simulere den proces en AI går igennem for at justere en weight til en passende værdi. Herunder er der beskrevet en grov gennemgang af kodens vigtigste pointer.



Figur 2 - Simpel visualisering af koden for weight adjustments

Inden vi beskriver koden, er der noget baggrundsviden man skal kende for at opnå den fulde forståelse. En OR activation funktion, som vi benytter her, tænker kun i binære tal, hvilket betyder at den kun kan modtage inputs af 1 og 0. Nedenstående har vi illustreret de mulige inputs og de tilhørende outputs, i formen input1 + input2 = output.

0 + 0 = 0

0 + 1= 1

1 + 0 = 1

1 + 1 = 1

Måden ANN finder ud af om den skal aktiveres, er med denne formel: $input1 * weight1 + input2 * weigth2 + bias > 0$(Ahirwar, 2017) Det betyder, at hvis den får et resultat over 0 vil den aktivere og sende sit output.

I begyndelsen af koden har alle vores variabler variabeltypen double, som er en variabeltype, der kan indeholde mange decimaler. Dette er nødvendigt, fordi vi senere i koden genererer en række tilfældige tal imellem -1 og 1. Dette tilfældige tal skal være så præcist som muligt, for den mest præcise repræsentation af virkeligheden. Derfor kræver det at vi bruger det maksimale antal decimaler vi har til rådighed. Disse tilfældige værdier bliver opgivet i funktionen InitialiseWeights.

I vores funktion 'Train' vælger vi hvor mange epochs vi vil benytte os af, jo flere gange man gennemfører jo større chance har man for at få et godt resultat, altså en AI som ikke

fejlvurderer. Selvom man kan træne den tusindvis af gange, giver det ikke altid bedre resultat end det som den kommer frem ved brug af få epochs Hvis den finder den korrekte weight i en tidlig gennemgang, får den ikke noget ud af gentage processen

Funktionen 'DotProductBias' er her hvor vi udregner de nye weights og bias med formlen der blev beskrevet tidligere under teorien i afsnit 2.2.2.3 om justering af weights.

Funktionen 'Updateweights' tjekker om det output vi har fået matcher med det som vi har oplyst og hvis ikke så justerer weighten.

Opsætningen i Unity er relativ simpel. Vi fortæller kun Unity hvad den skal behandle som de "rigtige svar". Dette rigtige svar følger OR activation function som blev beskrevet tidligere.

I bilag 2.2 ses en række tal som er taget fra konsollen i Unity. Her ses W1, W2 og B. Disse 3 variabler står for weight 1, weight 2 og bias. I de første 4 linjer står der 3 tilfældigt genererede tal. Disse tal er som nævnt tidligere tilfældige mellem -1 og 1. Ud fra disse tre weights så kontrollerer den sine resultater i forhold til OR activation functions og printer teksten "Total Errors:" herefter antallet af fejl som ANN har lavet.

Hvis der er registreret nogle fejl, går den koden igennem igen og vil nu printe de nye weights og bias. Testen, der står til sidst i konsollen, er en test og kontrol af, at OR activation funktionen virker.

6.2 Hvad har vi lært?

I dette afsnit forklares det, hvordan vores læringsproces har været i forhold til forståelsen af simulation og weights og hvilke processer, der som har været en del af at hjælpe denne proces på vej mod dets mål.

Vores simulering af et ANN, som justerer weights, har givet os et indblik i både hvordan weights opfører sig og hvordan de ændrer sig.

Før simuleringen var det især hvordan en weight opfører sig, som var en udfordring for os at forstå. Men efter vi selv har skrevet koden og haft muligheden for at kunne gå ind og kigge specifikt på weights, især med brugen af konsollen, som har hjulpet os med at forstå weights bedre.

Vi har gennem udarbejdningen af denne simulation lært om weight justering i forhold til hvordan en AI gør dette. Vi har også lært om forskellige activation functions, som er måde ANN beregner hvad outputtet skal være. Især brugen af en activation function har hjulpet os meget med forståelsen af disse. I forhold til vores hypotese blev det bekræftet til en vis grad. Det passede med at weights bliver påvirket og ændret til at passe til outputtet, til gengæld vidste vi ikke hvordan en activation function havde påvirkning på outputtet.

7. Produkt

I dette afsnit vil vi gennemgå vores produkt, herunder en forklaring på vores kode. Produktet er, som tidligere præsenteret, en virtuel simulation af en selvkørende bil.

Vi bestræber os på at gøre simulationen så realistisk som muligt, ved at fortælle bilen hvad den har mulighed for at gøre, så den agerer ligesom en bil og ikke bare stopper med det samme eller bare teleporterer bilen over til den ønskede destination. Vi vil generere en bane, som bilen kan køre på og vise den, at den skal igennem banen uden at ramme noget på turen.

Vores produkt ender ikke med at være en helt virkelighedstro selvkørende bil i en realistisk verden. Hvis vi skulle simulere bilen, med alle de forskellige situationer den kan komme ud for, ville det være for tidskrævende, til at vi kunne nå det. Af denne årsag har vi valgt at fokusere på de områder, som vi synes bedst repræsenterer, hvordan en selvkørende bil kører rundt. Måden hvorpå vi gør dette, er med et simpelt mechanical LiDAR system. Dette giver os muligheden for at følge med i hvordan bilen orienterer sig og hvordan den reagerer ud fra de inputs den får. Dertil er dette LiDAR system baseret på principperne bag en rigtig LiDAR-sensor, dog i en mindre og mere simpel udgave.

7.1 Indledning

I denne simulering tager vi, hvad vi har lært, om udvalgte sensorer og Artificial Neural Network (ANN), og bruger det til at lave en simuleret selvkørende bil. Simulationen skal fungere sådan, at en person producerer noget træningsdata ved at køre på banen. Træningsdata vil så blive gemt og den skal kunne hentes og læres af et ANN, som derefter skal være i stand til at kunne køre den selvsamme rute, eller en anden rute, uden at køre ind i noget.

Vores simulerede bil vil kun indeholde en primitiv version af en LiDAR-sensor, som den vil bruge som input. Samt en observering af hvorvidt bilen kører frem, tilbage, til højre eller venstre.

Under konstruktionen af denne simulation håber vi på at få et større indblik i hvordan en AI udnytter den data den får givet, samt hvilke mulige problemer simulationen kan støde på.

7.1.1 Hypotese

Vores hypotese for konstruktionen af denne simulation er, at AI'en efter en vis mængde træningsdata vil være i stand til at kunne køre banen, eller en anden vilkårlig bane, fejlfrit og uden større problemer. Vi forventer den data den indsamler vil være gode nok til at AI'en skaber nogle generelle regler, for hvordan den skal opføre sig overfor nogle tilfældige situationer. Disse regler vil altså være generelle, og den simulerede bil vil ikke udelukkende være i stand til at køre på den specifikke bane, som vi vil generere og træne den i.

7.2 Gennemgang af simulationen

Ud over arbejdet, der er udført i selve Unity, som primært har bestået at skabe terrænet, så har der også været udarbejdet to programmeringsfiler. Disse to koder hedder henholdsvis Drive og ANNDrive.

Drive filen bliver kun udført under træningsprocessen. Dette er koden, som vi bruger til vores træningsbil, hvis formål er at generere træningsdata for os. Denne kode er altså aldrig aktiv sammen med den selvkørende bil, da den kun har til opgave at generere nogle træningsdata og gemme disse i en mappe som vores Artificial Neural Network kan finde og uddrage nogle informationer fra.

Den anden kode, som vil blive beskrevet, hedder ANNDrive. Denne kode er det, som vil blive aktiveret, når vi bruger den selvkørende bil. Denne kode henter dataen fra vores ANN kode, som ikke vil blive beskrevet i dette projekt, ud over den teori der allerede er blevet nævnt i afsnit 2.2.2.

Ud over de to koder, som er beskrevet, så gør vores selvkørende bil også brug af tre andre filer, som er selve ANN'et i simulationen. Dog er disse tre filer blevet taget fra internettet (Byl, 2019), da det ville være for stor en opgave på for kort tid at programmere et

velfungerende ANN på den tid vi har haft til rådighed. Det eneste, der er relevant at vide om disse tre koder, er, at det er disse, som skaber de regler som ANNDrive køre efter.

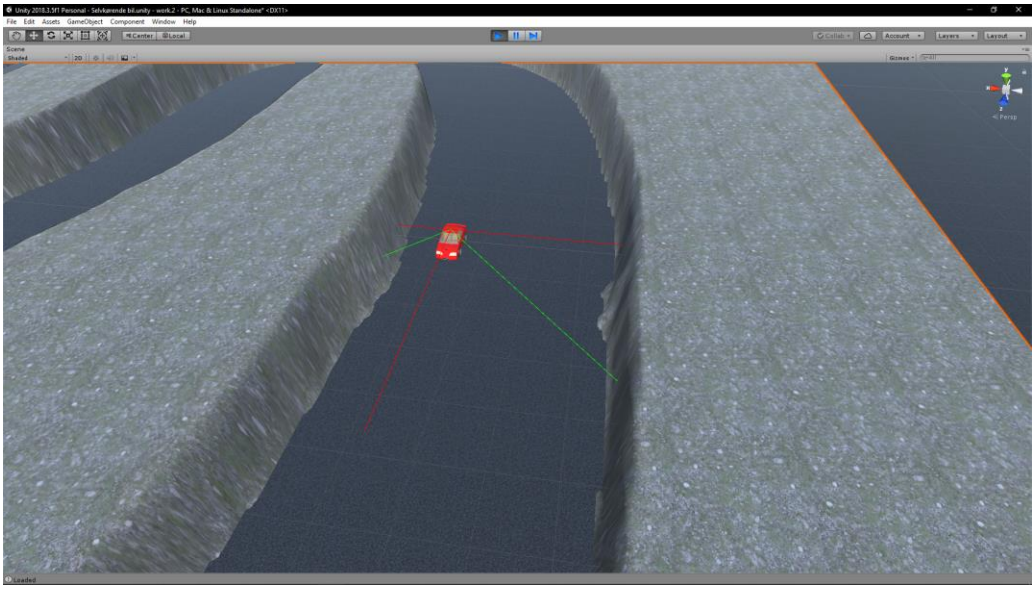
7.2.1 Drive file

Vi starter i denne kode, da det er her vi indsamler træningsdataen, som vi vil komme ind på i nedenstående afsnit.

Den første del af denne kode er standardkode for bevægelse af et objekt i et 3D miljø. Det eneste, der er interessant her, er, at man skal forstå at "Horizontal" og "Vertical" begge er et andet ord for tasterne "W", "S", "D" og "A" samt de fire piletaster, hvor "W", "S", op og ned piletasterne tilhører Horizontal, mens "D", "A", pil til venstre og pil til højre går ind under Vertical.

Efter denne del går vi ned og gør brug af kommandoen Debug.DrawRay. Det er denne kommando som vi vil gøre brug af under denne kode for at kunne tegne vores laser. Det DrawRay gør i praksis, er at lave en linje mellem to punkter. I vores simulation har vi valgt at lave denne linje, længden er 50 gange standardmålestokken, som Unity benytter sig af, ud fra bilens centrum. En af grundene til, at vi har en maksimal afstand, som laseren kan registrere, er, at en traditionel LiDAR-sensor også har en maksimal afstand den kan registrere. Så for at efterligne en LiDAR-sensor bedste muligt har vi givet vores laser dette maksimum. I vores simulation genereres der kun fem lasere, hvilket skyldes, at jo flere laserer vi genererer jo mere computerkraft skal der bruges for at AI'en kan lære af det. Dette er, som vi har nævnt under afsnit 5.2.1.1, ikke præcist som en LiDAR-sensor opererer i virkeligheden, men principperne bag en LiDAR-sensor er stadig eftervist i vores simulering.

Vi har valgt at lave fem lasere, som givetvis har retning fremad, til højre, til venstre, 45 grader til højre og 45 grader til venstre, da dette giver os det bedste billede af omgivelserne foran bilen. Der kunne her også have været en sensor bag på bilen, som ville registrer denne del, men da vores simulation ikke giver anledning til at bilen bakker, har vi ikke taget dette med. Nedenunder har vi taget et skærmbillede af vores bil, hvor vi har visualiseret disse fem lasere.



Figur 3 - Screenshot af simulationen i Unity

Disse lasere giver os en afstand på, hvornår de kolliderer med et andet objekt, i dette tilfælde væggene der er opstillet. Dette er hvad vi kalder dataen eller inputtet som AI'en får præsenteret. Disse inputs kan være et tal mellem 0 og vores visbleDistance, som er en variabel vi kan justere på. I vores tilfælde har vi givet vores visbleDistance en længde på 50. Hvis vi får et input der svarer til vores visbleDistance, betyder det, at vores DrawRay, altså laseren, ikke har kollideret med nogle objekter mellem dens startpunkt, ved bilen, og 50 ud eller til siden.

Dog skal det her bemærkes, at vi runder disse tal ned til et tal mellem 0 og 1 med et interval på 0,5, som det kan ses i koden med brugen af System.Math.Round. Dette gør vi for vores egen skyld, så vi får mere letlæselige tal.

Dette gør vi af flere grunde. Først runder vi disse tal ned, så vores ANN ikke skal bruge lige så meget computerkraft, når den skal generere nogle regler. Dog ville dette ikke være et problem i vores eksempel med en lille bane, men dette er noget man ville skulle tage hensyn til, hvis vi eksempelvis brugte samme kode ude i den virkelige verden hvor et ANN ville have mange flere inputs at skulle tage højde for.

En anden grund til at vi gør dette, er også, at på den måde kan vi lettere gå ind og ændre i vores træningsdata for det bedste resultat.

Den sidste grund til at vi runder tallene ned er, at vi som personer bedre ville kunne læse dem og derved lære af den data vi havde indsamlet. Det gør dataen mere overskuelig.

Den sidste del af denne kode er der, hvor vi gemmer denne data i en tekstfil, som senere kan blive læst af vores AI. Dataen bliver gemt i en lang string af data, som består af forskellige afstande. Disse afstande bliver adskilt af et komma, som senere hen også bliver brugt af AI'en til at se forskel på de forskellige inputs. Denne data vil blive gemt i en mappe, som vi i starten af koden skaber ved navn "trainingData.txt".

Til sidst i koden har vi et lille stykke kode, som vil kontrollere og formindske antallet af irrelevante inputs, som det ANN vil modtage i vores trainingData.txt. Dette betyder, at vi kontrollerer om den data vi skal til at importere til vores trainingData.txt text file ikke allerede findes derinde. Dette er ikke en stor fejlkilde, dog ville det gøre træningsdataen mere uoverskuelig for en person at læse. Det, at dataen allerede findes i trainingData.txt, betyder, at uden denne del af koden ville vores kode ved hver gennemgang af koden sende data til vores træningsmappe og denne gennemgang bliver fuldført 60 gange i sekundet. Problemet forekommer kun i nogle specifikke situationer, eksempelvis når programmet bliver startet. Bilen begynder muligvis ikke at kører i det øjeblik, som programmet starter. Dette betyder, at den nu vil optage en masse data, som vil se ens ud. Et andet eksempel kunne være, hvis bilen ikke flytter sig nok fra frame til frame, at forskellen ville blive registreret, når vores træningsdata bliver afrundet til 0, 0.1 og 1.

7.2.2 ANNDrive file

ANNDrive er koden til den selvkørende bil. I starten af denne kode konstaterer vi nogle variabler, heriblandt er de meste interessante visibleDistance, epochs, speed, SSE og lastSSE.

VisibleDistance og speed er begge nogle variabler, som vores Drive file også havde. Variablerne skal have samme værdi i ANNdrive, som de har i vores Drive file. Dette skyldes, at den data, som vi giver vores AI igennem filen trainingData.txt, skal passe med hvad vores AI faktisk ser. Hvis der var en forskel på de to sæt data, ville AI'en ikke kunne fungere optimalt, da de regler som den har lavet ud fra sin træningsdata, ikke ville være det samme, som hvad den har observeret i en gennemgang.

SSE står for sum of squared errors, som vi benytter for at vise hvor godt den beregner sine weights og så vi kan sikre os, at den ikke bliver dårligere til at køre, da vi har brug for nogle parametre for hvor gode de er. Hvis vi ikke målte dette, ville det ikke være muligt at forbedre den, da vi er nødt til at kunne fortælle den, hvad der er godt. Dette gør vi ved brug af en variabel kaldet LastSSE, som viser den forrige værdi af SSE vi havde. I tilfælde af at den

bliver højere end den nye SSE, sætter vi weights tilbage til den forrige, fordi den var bedre for vores selvkørende bil. Og så gør vi vores Alpha værdi lavere, for at få den til at bevæge sig i den rigtige retning. Men hvis den går den rigtige vej, så gør vi vores Alpha højere.

Alpha ligger under ANN og fungerer som en slags weight for ANN, da vi ganger med Alphaen. Dette hjælper os med at kunne nøjes med at benytte dele af vores træningsdata, så vi ikke bruger for meget. Eksempelvis hvis vi nu skal køre om et hjørne og dens træningsdata siger den skal mod venstre for ikke at køre ind i noget, så ville den kunne gå for langt mod venstre til hvad den ser som optimalt, fordi den bevæger sig en vis mængde og derfor kan gå for langt. Så kan vi sige, at vi bare vil bruge halvdelen af den afstand som den skal bruge. Så næste gennemgang vil den prøve at gøre det bedre, ved at ændre Alphaen for at få en bedre SSE.

Epochs er, som tidligere nævnt, antallet af gange som koden

blev trænet i vores program. En større mængde epochs er bedre for os, da vores SSE ikke ville kunne blive højere. Derfor ville vi kunne sætte antallet af epochs ligeså højt som vi har lyst til, hvilket vil give os et bedre resultat. Dog erfarede vi, at efter 1000 epochs skete der ikke særlig meget med antallet af SSE, så derfor har vi besluttet os for at beholde værdien på 1000.

I vores void Start har vi selve det ANN, som her bliver brugt. ‘ANN’ i koden er selve vores Artificial Neural Network, som kræver fem inputs, så den kan begynde at bearbejde vores træningsdata. Disse fem inputs er antallet af afstande den skal tage højde for, altså de fem drawRays (lasere på bilen), vi gør brug af for at indsamle træningsdata. Det næste input, som ANN kræver, er antallet af outputs, som der kræves af den. Dette er meget simpelt, da vi i vores tilfælde kun har to dimensioner den skal tage højde for, nemlig horisontalt og vertikalt. Disse to tal er også hvad vi kan se i vores træningsdata og er de eneste af de syv inputs, som vi giver vores ANN, som kan være et negativt tal. Her er positive inputs henholdsvis fremad og til højre, mens et negativt tal ville være tilbage og til venstre. Det næste input ANN kræver, er hvor mange hidden layers vi gør brug af. I denne simulation gør vi kun brug af ét hidden layer. Det næste input er antallet af neuroner vi gør brug af i hver af disse hidden layers. I vores tilfælde har vi brugt tallet 10 ud fra en tommelfingerregel, om at bruge dobbelt så mange neuroner som der er inputs. Dog ville AI’en også fungere med både mindre antal

neuroner eller flere. Kvaliteten af disse resultater kan variere. Det sidste input, som der bliver brugt, er hvad vores Alpha værdi skal starte på.

Efter disse fem inputs er opgivet til det ANN, starter vi selve læringsprocessen.

Læringsprocessen sker i LoadTraningSet og det er der, at det meste af udregningen i vores kode sker. Den starter med at finde filen med vores træningsdata, som tidligere er blevet genereret og indlæser dataen derfra. Efterfølgende laver vi et for loop med antal af epochs, og vi sikrer os den læser dokumentet korrekt.

Vores inputs for bevægelse er plads (5) og (6), når man kigger på den genererede træningsdata. Dog er det her vigtigt at vide, at disse inputs er gemt i en array, som gemmer sine informationer fra tallet 0 af. Dette betyder, at når vi henviser til informationen i vores kode så skal vi trække et fra placeringen af det omtalte input.

Det er i LoadTraningSet, at bilen skal reagere, og så længe plads (5) og (6) ikke er lige med 0, så skal den køre koden, hvor den sætter information ind i vores array fra (0) til (4). Så bruger vi funktion map til at ændre de værdierne som skal ind på plads (5) og (6) fra værdierne 0 og 1 fra dens forrige -1 til 1, hvilket betyder, at hvis den får -1 sætter den værdien til 0, 1 til værdien 1 og 0 til værdien 0.5 af samme grund som nævnt tidligere. Den beregner også vores SSE herinde og kontrollerer at den ikke blev højere, som også blev nævnt tidligere.

I update funktionen sker ikke særlig meget for os, fordi vi kører hele træningen inden vi starter og da processen kun skal køre en enkelt gang er der ingen grund til at havde den i update, men det er her bilen kører. De tal den har fundet frem til under processen sætter den ind her, og sammen med de nye informationer finder den ud af, hvordan den skal kører banen. Med map funktionen sætter vi den tilbage fra 0 eller 1 til -1 og 1, fordi AI'en skal bruge de rigtige tal. I Drive filen ændrede vi datasættet for at øge vores forståelse af datasættet.

Efter den har kørt sine epochs, vil den køre med de weigths, som den nu er kommet frem til under LoadTraningSet funktionen. For at slippe for at køre træningsprocessen gentagende gange, har vi valgt at gemme informationen, fra de weights den fandt frem til før, i et tekstdokument, ligesom vi gjorde med vores træningsdata. Dette giver os muligheden for at enten at køre programmet med den gemte fil eller skabe en ny. Dette sker i SaveWeight eller

LoadWeight funktionerne, som fungerer meget simpelt ved at loade en string, som den har gemt. Dette synes vi var vigtigt for vores simulation, da, hvis det skal forestille en virkelighedstro selvkørende bil, er det vigtigt, at den har dataen liggende og ikke skal simulere det hele inden den starter. Derudover betyder det også, at vi kan finde et godt sæt weights som fungerer til dels problemfrit.

Når vi gjorde det på denne måde, kunne vi køre et utal af epochs for at få de allerbedste weigths, hvilket vi havde gjort med 10000, for at se om det ville give et bedre resultat. Men da den gjorde det næsten lige så godt som ved 1000 epochs, valgte vi at sætte den tilbage til de 1000 epochs, så vi også havde mulighed for at eksperimentere på vores simulation uden en lang træningsproces og finde ud af hvad der skete hvis vi ændrede forskellige værdier.

Vi kom frem til efter en test, at vores ANN ikke umiddelbart ville blive bedre på grund af den øget mængde af epochs, dog observerede vi en større nøjagtighed til dens træningsdata.

OnGUI, som er en funktion vi benytter i vores kode, har ikke noget praktisk formål i vores kode. Vi bruger den kun for vores egen skyld, så det er nemmere at se om den virker efter hensigten, så vi kan debug hvis den ikke gør. Vi kunne have gjort det i konsollen, men vi følte det var mere overskueligt og nemmere for andre at se det også. Vi har kun valgt det vigtige for os, som er epochs, SSE og Alpha. Epochs så vi altid kunne se hvor langt den var i træningsprocessen og så vi kunne vælge et godt antal epochs ud fra SSE, som var vigtig så vi kunne se hvor godt vores weights blev beregnet og til sidst vores Alpha værdi.

Eftersom vi har valgt at benytte os af menneskeskabt data til at få vores selvkørende bil til at køre, vil dens potentiale på dette grundlag også blive begrænset. Da den rute vi tog under vores simulation, hvor vi genererede træningsdataen, vil være den rute som den prøver at genskabe, når den selv kører. Dette betyder, at hvis personen som generede træningsdataen havde en tendens til at tage skarpe sving og holde sig helt i højre side, vil den også gøre dette. Når det næsten er umuligt for et menneske at køre helt perfekt igennem banen, vil der altid være nogle fejl, som den vil genskabe. Hvis vi ønskede en selvkørende bil, som kørte perfekt, ville det være smartere at omskrive koden, så den ikke ville bruge vores træningsdata som målet, men mere som et springbræt, hvor en anden træningsmetode tog over, hvor den lærte fra sig selv efterfølgende.

Grunden til vi ikke havde lavet den sådan er, at den ville være svære for os at programmere og ville tage længere tid at træne den, så vi besluttede at lave den anden først da vi følte det

var vigtigere for os at havde en funktionelt selvkørende bil end en perfekt bil, så vi kunne eksperimentere med den og lære om den selvkørende bil i processen.

7.3 Opsamling

Vores hypotese blev til dels godtaget. Vores produkt endte med at køre på vores bane og den lavede nogle generelle regler som bilen kunne køre efter. Det vigtige med disse regler er at de ikke tog udgangspunkt i et specifikt stykke af banen, og derfor virker reglerne til alle typer baner eller veje hvilket var vores mål med produktet. Dog skal det nævnes, at banen og vejen skal have den samme bredde, som den bane vi har brugt til at producere dens træningsdata.

Vi fandt til gengæld ud af, at vores trænings data ikke altid ville være perfekt og derfor vil vi forkaste den del af vores hypotese som forventede at alle vores træningsdata altid ville give gode testresultater.

8. På hvilken måde har konstruktionen af en simulering bidraget til vores forståelse og læring af AI?

I det nedenstående afsnit vil der på baggrund af teorien *'learning by doing'* og Schöns refleksionspraksisser blive analyseret på, hvordan disse læringsmetoder har bidraget til vores forståelse og læring af AI.

I begyndelsen af projektet var alle seks gruppemedlemmer optimistiske omkring at lave en selvkørende bil. Den første tanke faldt på, at vi skulle lave en mindre prototype af en selvkørende bil, men efter nærmer overvejelse kom vi fra denne idé igen, på det grundlag at det ville blive for tidskrævende og for svært at vise hvordan selve AI'en fungerer. Derfor valgte vi at konstruere en simulation i Unity, hvor vi havde bedre mulighed for at præsentere og forstå de data som vores selvkørende bil indsamlede og- eller lærte. Den første indskydelse var, at det ville være relativt ligetil at lave en selvkørende bil i Unity, men efter at have haft dannet os nogle erfaringer, fandt vi hurtigt frem til, at det var mere kompleks og besværligt end først antaget. Grunden til dette kan være, fordi vi ikke har haft så meget knowing in action, da vi ikke førhen har arbejdet med selvkørende biler.

For at opbygge en forståelse, for hvordan vi skulle konstruere simulationen af den selvkørende bil, har vi reflekteret over vores fremgangsmåde. Derfor besluttede vi os for at

indsamle ny viden omkring emnet. Den nye viden blev fundet gennem relevante bøger og internetkilder og derudover deltog to fra vores gruppe i online kurser, som det kan læses i logbogen (bilag 4). Det var et kursus, der handlede om de basale ting i Unity, såsom programmeringssproget C#, hvilket vi ikke havde kendskab til førhen. Det andet online kursus handlede mere om hvordan Machine Learning fungerer. Her fandt vi ud af, at et Artificial Neural Network kan justere sine variabler. Til gengæld fandt vi ikke helt ud af hvordan den gjorde dette. Ved at forsøge og eksperimenter os frem til hvordan den justerede sin variabel i simulationen, fandt vi frem til at den gør det på den måde at den ændre sine weights. Efter vi havde erfaret os den nye viden fra dette online kursus, var vi nu i stand til at lave en simulering, hvor vi havde fået en forståelse af de weights og activation functions som finder sted i simuleringen. Dette er et klassisk tegn på knowing in action. AI er et relativt nyt emne for samtlige medlemmer i gruppen, så det er begrænset hvor meget tidligere viden og erfaring vi kan trække på, mens vi konstruerer simulationen. Derimod er vi alle blevet præsenteret for programmering, hvor vi implicit trækker, på de erfaringer vi har derfra. Ved hjælp af at have læst op på de forskellige teknologiske artefakter og former for Machine Learning, har vi fået en bedre grundforståelse for hvordan en selvkørende bil fungerer. Ved at have fulgt et online kursus, fik vi en "hands-on experience", og har derved opnået ny viden ved learning by doing. D. 12-5-2019 fik vi lavet vores første simulering af en selvkørende bil. Den første simulering vi fik lavet, var ikke helt optimal og virkede ikke helt som vi gerne ville have den til. Derfor lavede vi reflection on action, og her fandt vi ud af, at den bedste måde at gøre det på, var ved at opdele simulationen i to dele (bilag 4). Ved at opdele simulationen, menes der, at vi deler den op, så selve bilen står for sig selv, og de weights den generer også står for sig selv. Weights kan ses i bilag 2. Grunden til dette var, at det ville blive for svært både at skulle holde styr på weights, samtidig med vi havde fokus på bilen. Så efter refleksionen valgte vi at have mere fokus på bilen, og så finde inspiration til weight adjustment fra internettet.

Det viste sig at være en god ide at flytte vores fokus fra weight adjustment til et fokus på at få simuleret den selvkørende bil i Unity. Efter et par dage havde vi fået opstillet en selvkørende bil, som var i stand til at køre via et Artificial Neural Network, med inputs fra DrawRays. I Unity opfører de sig ligesom laserne fra en LiDAR, tilkoblet en selvkørende bil. Ved at reflektere over arbejdsprocessens størrelse, med hensyn til hvis vi skulle holde et lige så stort fokus på weight adjustment, som vi har gjort på simuleringen af den selvkørende bil, ville det blive for tidskrævende. Så ved at reflektere over vores handlinger, kom vi frem til en løsning,

der sikrede os at vi kom videre med projektet. Reflection in action, kommer ikke så meget til udtryk i vores logbog, men har været en del af læringsprocessen i Unity. Her er den kommet til udtryk, når vi skulle ændre noget i vores simulation. Det kunne eksempelvis være, hvis vi gerne ville have bilen til at køre uden om et objekt på banen, som den ikke har stødt på før, men hvor vi ikke ved hvad resultatet vil ende med. Der havde vi en erfaring, hvor vi satte en firkant midt på banen, og gav bilen nogle nye træningsdata, ved at testkøre banen som vi gerne ville have det. Resultatet endte med, at den kørte uden om det nye objekt, men til gengæld havde den oprettet nogle nye regler, hvor at i stedet for at kører midt på vejen, ville bilen helst køre så tæt på muren som overhovedet muligt. Dette resultat var ikke helt meningen. Efter vi lavede reflection on action på denne nye erfaring, kom vi frem til at det formentlig nok var fordi at firkanten og væggen var lavet af det samme materiale/stof. Vi tænkte nu på ny, hvordan vi så kunne få det ønskede resultat. Ved at lave objektet og væggen i forskelligt materiale, fik bilen nemmere ved at adskille dem fra hinanden og kørte uden om objektet uden at lave nye regler, om at den helst skal køre så tæt på væggen som muligt.

Hvilken type AI fungerer bedst?

Til at starte med vidste vi ikke hvilken slags AI, der ville fungere bedst i vores simulation. Til at starte med havde vi både reinforced learning, unsupervised learning og supervised learning i betragtning, selvom vores første indskydelse, inden vi testede dem, var, at reinforced learning ville fungere bedst. Det blev baseret på det grundlag, at med reinforced learning ville den selvkørende bil være i stand til at køre i baggrunden og selv finde ud af de handlinger, som ville give den flest points. Men efter nogle forsøg med reinforced learning, stod det klart for os, at programmering af en selvkørende bil baseret på reinforced learning ville blive for teknisk svært, på det stadie vi er nu. Efter vi havde reflekteret over reinforced learning, kom vi frem til at forsøge med unsupervised learning i stedet. Med unsupervised learning opfører den sig mest menneskeligt, hvilket skulle være argumentet for at benytte denne. Derimod, som nævnt i teorien, bliver der dannet et black box, hvor vi ikke kan se hvad den finder frem til. Black boxen ville vi ikke lære særlig meget nyt af, og vi fandt også frem til at unsupervised learning var lige så teknisk svært som reinforced learning.

Den sidste Machine Learning vi havde tilbage var supervised learning. Vi fandt hurtigt ud af at denne måde var markant nemmere at programmer end de andre, på vores nuværende niveau og dermed også bidrog mere til vores forståelse af hvordan AI fungerede i en selvkørende bil. Modargumentet for supervised learning er, at man formentlig ikke ville

benytte denne Machine Learning i en virkelig selvkørende bil, på det grundlag af, at man aldrig kan vise den selvkørende bil, alle de uendelige mængder scenarier man kan komme ud for på vejene. Men taget det i betragtning valgte vi alligevel supervised learning, da det efter vores erfaring var det vi lærte mest af og det vi kunne komme længst med i forhold til vores produkt.

8.1 Delkonklusion

Vi kan altså konkludere at konstruktionen af den selvkørende bil i Unity har haft en betydning i forhold til vores forståelse og læring af Artificial Intelligence. Først og fremmest har det været med til at give en forklaring og simplificeret de ting vi har lært man kan benytte, men som vi ikke forstod hvordan fungerede. Derudover har det været med til at sætte tingene i perspektiv i forhold til hvad der er realistisk at nå, samt hvor kompleks og besværligt AI kan være. Til sidst har den givet svar på hvilken slags AI, der fungerede bedst i vores tilfælde, hvor vi fandt frem til, at supervised learning får den til at køre bedst muligt i vores simulation.

9. Diskussion

I dette afsnit vil vi diskutere følgende:

9.1 Hvordan lærte vi bedst?

I dette afsnit vil vi diskutere vores designproces, samt inddrage fordele og ulemper mellem forholdet af hvad man kan lære af at læse og afprøve det, baseret på de erfaringer vi har gjort under processen.

Fra starten af projektet til slutningen af projektet, har designprocessen foregået på følgende måde.

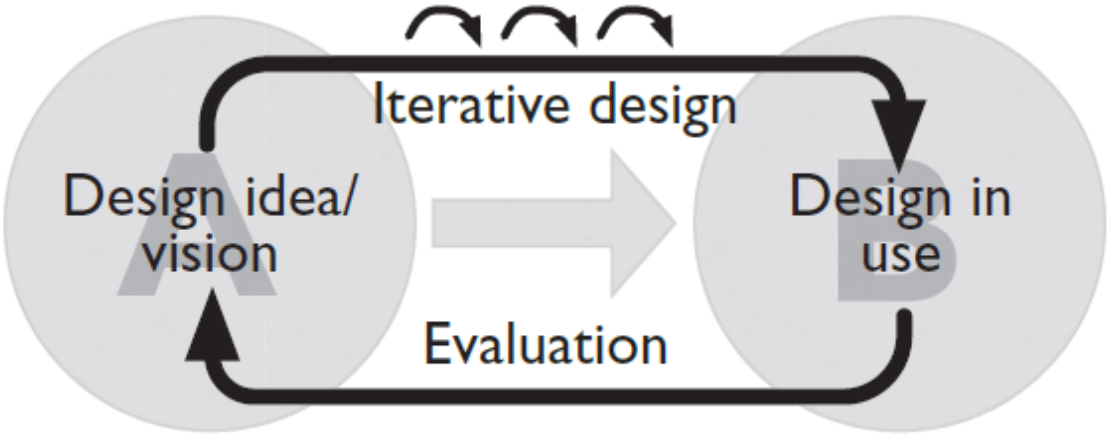


Figure 14.2 The design process depicted as a series of iterations. © Jesper Simonsen 2010.

Figur 14.2 (Simonsen, Bærenholdt, Büscher & Scheuer, 2010; s. 205)

Vi startede med at overveje hvilke designs, der kunne være relevante og hvad det er vi gerne vil komme frem til. Her ville vi gerne opnå en bedre forståelse af en selvkørende bils AI og teknologiske artefakter. Vi kom frem til, at vores produkt skulle være en selvkørende bil i Unity, der fungerer på de samme præmisser som en virkelig. Derefter har vi fundet relevant viden og litteratur, med det formål at få skabt vores produkt. Den viden har vi så eksperimenteret med og inddraget i vores produkt. Når vi så har et produkt, så gennemgås hele den ovenstående model igen, så det bliver en iterativ proces. Ved at lave denne iterative proces skulle det ende ud i det bedst mulige slutprodukt for os. Både 'learning by doing' og den traditionelle læringsmetode ved at læse, har været de to gennemgående læringsmetoder i designprocessen. I processen har vi sammenkoblet de to læringsmetoder, med forhåbningen om at det ville give den bedste forståelse af den AI og teknologiske artefakter en selvkørende bil bruger.

Baseret på vores erfaringer, hvis man kun tager den traditionelle læringsmetode, er den rigtig god til at give en grundforståelse for de forskellige former for Machine Learning, der kan benyttes i en selvkørende bil. Det var også denne fremgangsmåde, vi benyttede til at finde frem til, hvordan de forskellige teknologiske artefakter fungerede hver for sig. Der er ingen tvivl om, at den traditionelle læringsmetode også har bidraget til vores læring og forståelse, men som nævnt i analysen, fik vi ikke den dybe forståelse af det. Vi havde stadig en del tvivlsspørgsmål omkring hvordan den justerede sine weights og hvilke data softwaren i den selvkørende bil outputtede.

Tager vi kun udgangspunkt i John Deweys teori, var det rigtig svært at komme i gang uden nogen grundforståelse. I det tidlige stadie af projektet var vi stadig optimistiske omkring at få lavet simulationen, men det var først efter vi havde fået læst en masse litteratur og deltaget i de to online kurser, at vi rigtig kom i gang med konstruktionen. Herimod gav konstruktionen os en meget bedre forståelse af det vi havde læst. Det blev mere overskueligt at følge med i hvordan Machine Learning fungerede, eftersom vi kunne følge med i hvordan den dannede dens data, baseret på den testrunde vi selv havde kørt. Det svarede også på hvilken Machine Learning der fungerede bedst, i vores tilfælde, som nævnt i analysen, var det supervised learning. Derimod vil vi også argumentere for, at den nye viden vi har opnået, sidder bedre på vores hukommelse, eftersom vi har fået dannet nogle minder ved at konstruere det selv.

Baseret på vores erfaringer er de to læringsmetoder fornuftige nok i selv og de kan sagtens stå alene. Men derimod fandt vi frem til, at i vores tilfælde fungerede de bedst i sammenspil. Først gå ud at skabe en grundviden, derefter eksperimentere og lege med den nye viden. Her fandt vi også frem til, at man lærer rigtig meget af at fejle med eksperimenterne, på det grundlag, at man så bliver nødt til at tænke nye veje, man kan gøre det på.

9.2 Læring af AI

Der bliver i vores analyseafsnit nævnt noget omkring vores valg af brugen af supervised learning. Grunden til at vi ikke fik en simulation, baseret på reinforced learning, skyldes, at det er langt sværere at konstruere end supervised learning. Vi var nødt til at erkende, at det ville være at gabe over mere end vi kunne håndtere. Men dette betyder ikke, at vi ikke er tilfredse med vores slutprodukt. Vi lærte om AI, og et udpluk af de forskellige former for Machine Learning, samt hvilke egenskaber de hver især tilbød.

Vi opsatte nogle mål og forventninger, til hvad vores simulation skulle kunne, og prøvede derfor at efterkomme dem. Som sagt lykkedes det ikke 100% Men igennem denne proces lærte vi om de forskellige typer AI, og vi lærte at nogle typer var mere komplicerede end andre. Vi kom dermed frem til, at AI og Machine Learning kan forstås på flere måder, og man kan gå mere i dybden med dem på flere niveauer, alt efter hvor meget erfaring man har med AI. For at arbejde med reinforced learning, havde vi været nødt til at starte fra bunden af. Derfor tog vi udgangspunkt i supervised learning, hvilket var lettere at skabe nogle regler for, og lod vores læring starte derfra. Ud fra Donald Schöns tre læringsmetoder, har vi nu

næste gang vi arbejder med AI, muligheden for at gøre brug af hans "knowing in action", eftersom vi ikke længere er helt nye på emnet. Vi behøver ikke at starte med at undersøge alle de forskellige typer af AI og Machine Learning i samme grad, men vi kan nu fokusere på at gå længere i dybden med dem.

10 Konklusion

Vi har gennem denne rapport forsøgt at svare på følgende problemformulering: Hvordan kan vi gennem konstruktionen af en simulation undersøge og forstå AI og udvalgte teknologiske artefakter i en selvkørende bil?

Gennem rapporten kom vi frem til nogle mindre konklusioner, heriblandt at produktionen af selvkørende biler, stammer helt tilbage til 1926 og der siden dengang er sket en stor udvikling, især inden for de sidste 10 år, hvor Waymo har produceret den selvkørende bil tættest på level 5.

Vi kunne tidligt i processen allerede konkludere, at det ikke var tilstrækkeligt kun at benytte konstruktionen af simulationen, som baggrund for læring, da emnet var mere komplekst end først forventet. For at forstå hvordan AI og udvalgte teknologiske artefakter fungerede, blev vi nødt til at finde viden om det gennem relevant litteratur. Gennem den nyfundne litteratur, kunne vi konkludere at selvkørende biler gør brug af flere forskellige teknologiske artefakter heriblandt, sensorer, kameraer og AI. Hvert enkelt artefakt har en betydelig rolle, i det teknologiske system, der får den selvkørende bil til fungere. Med denne nye baggrundsviden, var vi klar til at gå i gang med simulationen. Under konstruktionen af simulationen havde vi især fokus på LiDAR-sensor og AI, både fordi det var to af de mest centrale teknologiske artefakter og fordi vi stadig havde nogle tvivlsspørgsmål omkring disse. Her kunne vi konkludere, at konstruktionen baseret på John Deweys teori *'Learning by doing'* og Schöns refleksion praksisser, gav en bedre og dybere forståelse, for det vi gerne ville finde ud af. Det gjorde det blandt andet fordi, at vi kunne følge med i hvordan AI fungerede i praksis, heriblandt hvordan den opsætter regler og justerer sine weights. Alt i alt gav konstruktionen og det færdige produkt, et mere overskueligt billede af hvordan de hver især fungerede og hvordan de sammenspiller. Til gengæld havde konstruktionen ikke en ligeså stor effekt som vi havde håbet, vi måtte erkende at forskellige typer af Machine Learning, var for teknisk

svært for os at benytte, så der har konstruktionen ikke bidraget til andet end det var meget kompleks AI.

Litteraturliste

Bøger

Flach, P. (2012) *Machine Learning: The art and science of algorithms that make sense of data*. Cambridge, England: Cambridge University Press.

Fujii, T. & Fukuchi, T. (2005) *Laser Remote Sensing.* Boca Raton, USA: Taylor & Francis Group.

Kolstrup, S. (2002) *Demokratisk dannelse - udstødt af den pædagogiske forskning* (2. udgave). Odense, Danmark: Syddansk universitet

Picton, P. (2000) *Neural networks* (2. udgave). Basingstoke, England: Palgrave.

Rogers, E. M. (2003) *Diffusion of Innovation* (5. udgave). New York, USA: Simon & Schuster.

Russell, S. J. & Norvig, P. (2010) *Artificial Intelligence: A Modern Approach* (3. udgave) Upper Saddle River, USA: Pearson Education.

Schön, D. A. (2008). *The Reflective Practitioner: How Professionals Think In Action*. New York, USA: Basic Books.

Simonsen, J., Bærenholdt, J. O., Büscher, M., & Scheuer, J. D. (red.) (2010). *Design Research: Synergies from Interdisciplinary Perspectives.* [Billede] England, London: Routledge.

Watzenig, D., & Horn, M. (2017). *Automated Driving: Safer and More Efficient Future Driving*. Schweiz: Springer International Publishing.

Wilson, R. A. & Keil, F. C. (1999). *The MIT Encyclopedia of the Cognitive Sciences*. Cambridge, USA: MIT Press.

Internetkilder

Ahirwar, K. (2017, 1 November). *Everything you need to know about Neural Networks* [Formel] Lokaliseret 12 Maj, 2019, fra https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491

Byl, P. d. (2019, Marts). *A Beginner's Guide To Machine Learning with Unity*. Lokaliseret 15 Maj, 2019, fra https://www.udemy.com/machine-learning-with-unity/

Danmarks vindmølleforening. (2013, Maj). *Sådan fungerer en vindmølle: Faktablad T1*. Lokaliseret 17 Maj, 2019, fra http://www.dkvind.dk/fakta/T1.pdf

Ejlskov, J., & Pindstrup, L. (2019, 19 Januar). *Learning by Dewey – del 2: Hvad skal vi lære?* Lokaliseret 12 Maj, 2019, fra http://www.lektionen.dk/blog/19/1/2017/learning-by-dewey-del-2

Hecht, J. (2018, Januar) *Lidar for Self-Driving Cars*. Lokaliseret 16 Maj, 2019, fra https://www.osa-opn.org/home/articles/volume_29/january_2018/features/lidar_for_self-driving_cars/

Jørgensen, N. (29, 10, 2018). *Teknologiens indre mekanismer og processer. Eksemplificeret ved digital signatur.* [PDF] Lokaliseret d. 22 Marts, 2019, fra https://moodle.ruc.dk/mod/resource/view.php?id=200066

Khader, M., & Cherian, S. (2018, Oktober). *An Introduction to Automotive LIDAR.* Lokaliseret 16 Maj, 2019, fra http://www.ti.com/lit/wp/slyy150/slyy150.pdf

Larsen, S. (2007, 2 Marts). *John Dewey - En introduktion*. Lokaliseret 13 Maj, 2019, fra https://www.folkeskolen.dk/45474/john-dewey---en-introduktion

Lekach, S. (2018, 16 Oktober). *Dubai launches self-driving taxi service*. Lokaliseret 16 Maj, 2019 fra https://mashable.com/article/dubai-rta-self-driving-taxi-cars/?europe=true

Neal, A. (2018, 24 April). *LiDAR vs. RADAR* [Billede]. Lokaliseret 16 Maj, 2019, fra https://www.sensorsmag.com/components/lidar-vs-radar

Olson, E. (2018, 30 Oktober). *How Does GPS Work?* Lokaliseret 14 Maj, 2019, fra https://insights.globalspec.com/article/10315/how-does-gps-work

Reed, D., & Widger, D. (2008, 1 August). *Democracy and Education, by John Dewey*. Lokaliseret 13 Maj, 2019, fra http://www.gutenberg.org/files/852/852-h/852-h.htm

Social- og Sundhedsskolen Viborg Amt. (s.d.). *vejledning til logbogsskrivning.* Lokaliseret 1 Maj, 2019, fra www.portoeruddannelse.dk/media(256,1030)/Vejledning_til_logbogsskrivning.dot

Waymo LLC. (2018). *Journey – Waymo*. Lokaliseret 14 Maj, 2019, fra https://waymo.com/journey/

Videnskabelige artikler

Bimbraw, K. (2015). *Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of*

Autonomous Vehicle Technology. Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics, , 191–198. https://doi.org/10.5220/0005540501910198

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). *Reinforcement Learning: A Survey.* Journal of Artificial Intelligence Research 4, , 237–285. Lokaliseret fra https://www.jair.org/index.php/jair/article/view/10166/24110

Kotsiantis, S. B. (2007). *Supervised Machine Learning: A Review of Classification Techniques.* Informatica 31. 249–268. Lokaliseret fra http://www.informatica.si/index.php/informatica/article/viewFile/148/140

Lim, B. S., Keoh, S. L., & Thing, V. L. L. (2018). *Autonomous vehicle ultrasonic sensor vulnerability and impact assessment.* 2018 IEEE 4th World Forum on Internet of Things (WF-IoT). Lokaliseret fra http://eprints.gla.ac.uk/153507/13/153507.pdf

Maan, A. K., Jayadevi, D. A., & James, A. P. (2017). *A Survey of Memristive Threshold Logic Circuits.* IEEE Transactions on Neural Networks and Learning Systems, 28(8), 1734–1746. https://doi.org/10.1109/tnnls.2016.2547842

Mikolov, T., Karafiat, M., Burget, L., Cernocký, J. & Khudanpur, S. (2010) *Recurrent neural network based language model.* Lokaliseret fra https://www.isca-speech.org/archive/archive_papers/interspeech_2010/i10_1045.pdf

Panchal, G., Ganatra, A., Kosta, Y. P., & Panchal, D. (2011). *Behaviour Analysis of Multilayer Perceptrons with Multiple Hidden Neurons and Hidden Layers.* International Journal of Computer Theory and Engineering, 3(2), 332–337. Lokaliseret fra http://www.ijcte.org/papers/328-L318.pdf