

---

# Inhoudsopgave

<b>H1 Voorwoord</b>	<b>5</b>
<b>H2 Inleiding en onderzoeksvraag</b>	<b>6</b>
<b>H3 Theoretische achtergrond</b>	<b>8</b>
3.1 Basis van programmeren	8
3.1.1 Programmeertalen	9
3.1.2 Het nut van programmeertalen	10
3.1.3 Soorten programmeertalen	11
Low Level programmeertalen	12
Medium Level programmeertalen	13
High Level programmeertalen	14
3.1.4 IDE's	15
3.1.5 Compilers	18
Optimalisatie	19
Soorten bugs	21
3.1.6 ASCII	22
3.2 Programmeertalen	24
3.2.1 C, C++	24
3.2.2 Python	25
3.2.3 HTML, CSS en JavaScript	26
3.2.4 API en Libraries	28
3.2.5 TensorFlow	29
3.3 De Hardware	30
3.3.1 Raspberry Pi	30
3.3.2 Espressif ESP32	31
3.3.3 YDLidar X4	32
3.3.4 Pinouts, wirings	33
3.4 Artificial Intelligence	36
3.4.1 Neat Algoritme	36
3.4.2 Backpropagation	37
3.4.3 A.I. Hiërarchie	38

---

3.4.4 A.I. In de auto industrie	39
3.5 Neuraal Netwerk	40
3.5.1 Layers	40
Sigmoïdefunctie	41
Layer abstraction	43
3.5.2 Lokale optima	45
<b>H4 Deelvragen</b>	<b>46</b>
4.1 Kunstmatige Intelligentie	46
4.1.1 Wat houdt Artificial Intelligence in?	46
4.1.2 Wat zijn de voordelen van zelfrijdende auto's?	46
4.1.3 Welke methoden van dataverzameling zijn het meest geschikt voor zelfrijdende auto's?	47
4.1.4 Welk library gebruiken wij voor het opbouwen van ons neurale netwerk?	48
4.1.5 Hoe worden de beste auto's geselecteerd?	48
4.1.6 Hoe slaan wij het neurale netwerk op om de auto te laten rijden?	49
4.1.7 Hoe bepalen wij de structuur van het neuraal netwerk?	49
4.1.8 Hoe kunnen wij ervoor zorgen dat de zelfrijdende auto op een veilige manier rijdt?	50
4.1.9 Kan de A.I. zijn omgeving begrijpen en herkennen?	50
Nvidia model	51
Image preprocessing	54
Maken van het model	57
4.2 Simulatie	58
4.2.1 Welke programmeertaal gebruiken wij voor de simulatie?	59
4.2.2 Hoe wordt de auto bestuurd?	59
4.2.3 Hoe hebben wij de wereld opgebouwd?	62
4.2.4 Hoe wordt de informatie opgeslagen?	62
4.2.5 Hoe lezen we de data op?	64
4.3 Auto	66
4.3.1 Welke auto gaan wij gebruiken voor ons project?	66
4.3.2 Hoe zorgen wij ervoor dat de Lidar-sensor, Raspberry Pi en de motor genoeg stroom hebben in de auto?	67
4.3.3 Hoe geven wij voltages met de Raspberry Pi door aan de motor en servo van de auto?	68

---

4.3.4 Hoe sluiten wij de Lidar-sensor aan op de Raspberry Pi?	70
4.3.5 Hoe kunnen wij op afstand met de auto communiceren?	71
4.3.6 Welke programmeertalen gebruiken wij op de Raspberry Pi?	72
4.3.7 Hoe krijgen wij afstandspunten van de Lidar-sensor en geven wij deze door aan de kunstmatige intelligentie?	73
4.3.8 Hoe worden de lidar sensor, de camera en de console controller ingesteld?	74
4.3.9 Hoe kunnen wij verbinding maken met de Lidar en de controller?	76
4.3.10 Hoe controleren wij de auto en geven wij commando's?	77
4.3.11 Hoe wordt de TensorFlow Lite library ingesteld?	78
4.3.12 Hoe wordt het beeld van de camera omgezet naar een format dat verwerkt kan worden door de A.I.?	78
4.3.13 Hoe wordt de data van de Lidar gefilterd op basis van de gegeven hoek?	79
4.3.14 Hoe verzamelen we data om de Artificial Intelligence te trainen?	79
4.3.15 Hoe laten we de Artificial Intelligence de auto besturen?	80
4.4 Werkelijkheid	83
4.4.1 Wat willen wij onderzoeken met ons experiment?	83
4.4.2 Hoe gaan wij het onderzoek uitvoeren?	84
<b>H5 Conclusie</b>	<b>85</b>
5.1 Simulatie	85
5.2 Artificial Intelligence	85
<b>H6 Discussie</b>	<b>85</b>
<b>H7 Nwoord</b>	<b>87</b>
7.1 Broncode	87
<b>Appendix</b>	<b>88</b>
A.1 Logboek	88
A.2 Fotos	90
A.3 Onderdelen, Kosten	92
<b>Bronnenlijst</b>	<b>93</b>
Libraries	94
Auto	94
Kunstmatige Intelligentie	94
Simulatie	95

---

## H1 Voorwoord

We hebben A.I. als onderwerp gekozen, omdat het ons al lang interesseert. Het was voor ons van te voren niet duidelijk hoe het exact werkte, maar de mogelijkheden ervan leken oneindig. Verder dachten wij dat het een relatief nieuwe ontdekking was, waarvan nog niet alle mogelijkheden bekend zijn. Daarom wilden wij A.I. verder onderzoeken.

Nu weten wij dat A.I. niet zo nieuw is als wij van tevoren dachten en dat het al bestaat sinds 1956. De mogelijkheden zijn daarom al grotendeels verkend en het komt zelfs al voor in het dagelijks leven. Voorbeelden daarvan zijn: aanbevelingen op social media, video games of zelfrijdende auto's. A.I. heeft zijn potentie ook laten zien toen het schaakprogramma Alpha Zero werd gemaakt. Dit is een A.I. die beter schaakt dan de wereldkampioen schaken. Het was dan ook tijdelijk de beste schaakcomputer ter wereld.

Verder bestaat het plan om een A.I. te maken, die volledig zelfstandig een auto kan besturen. Hierbij ben je dus volledig afhankelijk van de AI. Het is hierbij van groot belang dat de A.I. voor elke mogelijke situatie getraind is. Als een A.I. een bijzondere situatie namelijk nog nooit gezien heeft, heeft het ook niet geleerd hoe het hierop moet reageren. Wie weet wat een A.I. zou doen, wanneer een vogel voor een van de sensoren langs vliegt.

Daarom leek het ons interessant om te onderzoeken hoe betrouwbaar een A.I. nou precies is. Maar ook hoe het exact werkt en hoeveel potentie het eigenlijk echt heeft. We horen eigenlijk alleen maar van wat A.I.'s bereikt hebben, maar hebben nooit echt de A.I. zien leren en weten hoe hij leert.

---

## H2 Inleiding en onderzoeksvraag

A.I. begint steeds vaker terug te komen in ons leven, het komt voor bij virtuele assistenten, video games, fraude detecties en zelfrijdende auto's. Wij verwachten dat in de komende vijf jaar A.I. nog meer toegepast zal worden in de auto-industrie. Maar hoe werken zelfrijdende auto's precies en hoe wordt er gebruikgemaakt van A.I. in het ontwikkelen van zelfrijdende auto's? Om daarachter te komen willen wij graag proberen een zelfrijdende auto (robot) te maken.

Onze onderzoeksvraag is daarom ook: hoe kunnen wij een auto zelfstandig in een simulatie een willekeurig parcours steeds beter laten rijden en dat overbrengen naar de realiteit? Om onze onderzoeksvraag te kunnen beantwoorden hebben wij meerdere deelvragen bedacht en uitgezocht.

Zelfrijdende auto's maken gebruik van hoge resolutie camera's en sensoren om een nauwkeurig beeld te creëren van de omgeving van de auto. Hierdoor kunnen ze zien wat er ver vooruit op de weg staat. Zo kunnen ze bewuste beslissingen nemen en zelfstandig rijden, zonder de besturing van mensen.

De veiligheid en zelfstandigheid van de zelfrijdende auto's hangt af van de hoeveelheid voortgang die de onderzoekers gemaakt hebben bij het ontwikkelen van de geschikte en betrouwbare AI. Onze verwachting is dat in de komende vijf jaar, A.I. in combinatie met andere sensoren zoals Lidar en Ultrasonic afstandsmeeteters zal worden toegepast.

Wij gaan onderzoeken hoe wij een auto (robot) zelfstandig in een willekeurig parcours steeds beter kunnen laten rijden in een simulatie en of wij dit kunnen overbrengen naar de realiteit. Om hierachter te komen gaan wij eerst onze deelvragen beantwoorden.

Om onze eerste deelvraag (Wat houdt A.I. in?) te kunnen beantwoorden gaan wij onderzoek doen op het internet en in onze omgeving en deze informatie combineren met de kennis waarover wij al beschikken.

Voor onze tweede deelvraag (Wat zijn de voordelen van zelfrijdende auto's?) gaan wij onze huidige kennis combineren met onderzoek op het internet.

---

Hetzelfde geldt eigenlijk ook voor onze derde (Welke methoden van dataverzameling zijn het meest geschikt voor zelfrijdende auto's?), vierde (Welk library gebruiken wij voor het opbouwen van ons neuraal netwerk?), vijfde (Hoe worden de beste auto's geselecteerd?), zesde (Hoe slaan wij het neuraal netwerk op om het auto te laten rijden?), zevende (Hoe bepalen wij de structuur van het neuraal netwerk? En achtste (Hoe kunnen wij ervoor zorgen dat de zelfrijdende auto op een veilige manier rijdt?) deelvragen, ook voor deze deelvragen gaan wij onderzoek doen op het internet en in onze omgeving en deze informatie combineren met de kennis waarover wij al beschikken.

Verder gaan wij een robot maken en deze robot een aantal proeven laten doen. We willen de robot een parcours uit laten voeren, een zelfgemaakt spel laten spelen en de robot in bijzondere situaties zetten om te zien hoe hij daarop reageert. Het doel van deze proeven is het simuleren van echte situaties, om zo te onderzoeken of het mogelijk is voor de realiteit. Onze hoofdvraag hopen wij uiteindelijk te kunnen beantwoorden met behulp van alle informatie die wij hebben verzameld door middel van onze deelvragen en ons experiment.

Wij verwachten dat deze hoofdstukindeling de meest logische manier is om onze onderzoeksvraag te beantwoorden, omdat op deze manier wij door middel van de deelvragen steeds meer informatie krijgen die wij nodig hebben voor het uitvoeren van ons experiment. Als wij ons experiment hebben uitgevoerd, hebben wij als het goed is alle nodige informatie verzameld om onze onderzoeksvraag te kunnen beantwoorden.

---

## H3 Theoretische achtergrond

In dit hoofdstuk wordt alle informatie die in dit PWS wordt geraadpleegd uitgelicht. Basisconcepten van Computer Science worden hierbij uitgebreid besproken, zoals hoe computers werken, het programmeren, het gebruik van libraries, kunstmatige intelligentie, neuraal netwerk en de huidige toepassingen daarvan in de auto-industrie. Daarnaast wordt er ook de wiskunde theorie die van toepassing is op kunstmatige intelligentie uitgelicht.

### 3.1 Basis van programmeren

De eerste computers werden voornamelijk gebruikt voor eenvoudige berekeningen. Aangezien alle gegevens numeriek worden opgeslagen, realiseerden mensen zich al snel dat computers in staat zijn om ingewikkelde berekeningen uit te voeren. Dankzij de hoge snelheid van moderne computers, is de nauwkeurigheid van voorspellingen vergroot en de rekentijd verminderd. Dit maakt computers geschikt voor zaken waarin kunstmatige intelligentie toegepast kan worden, zoals zelfrijdende auto's en robotchirurgie. Omdat computers ook in kleine en goedkope varianten beschikbaar zijn, worden ze in veel zaken gebruikt zoals IoT, het slim maken van alledaagse apparaten en weersvoorspellingen.

Programmeren is het schrijven van computerprogramma's. Het wordt een steeds belangrijkere vaardigheid voor de maatschappij, aangezien de functionaliteit van computers wordt bepaald door het goed functioneren van het programma. Zonder een programma zijn computers slechts gecompliceerde machines die elektriciteit in warmte omzetten.

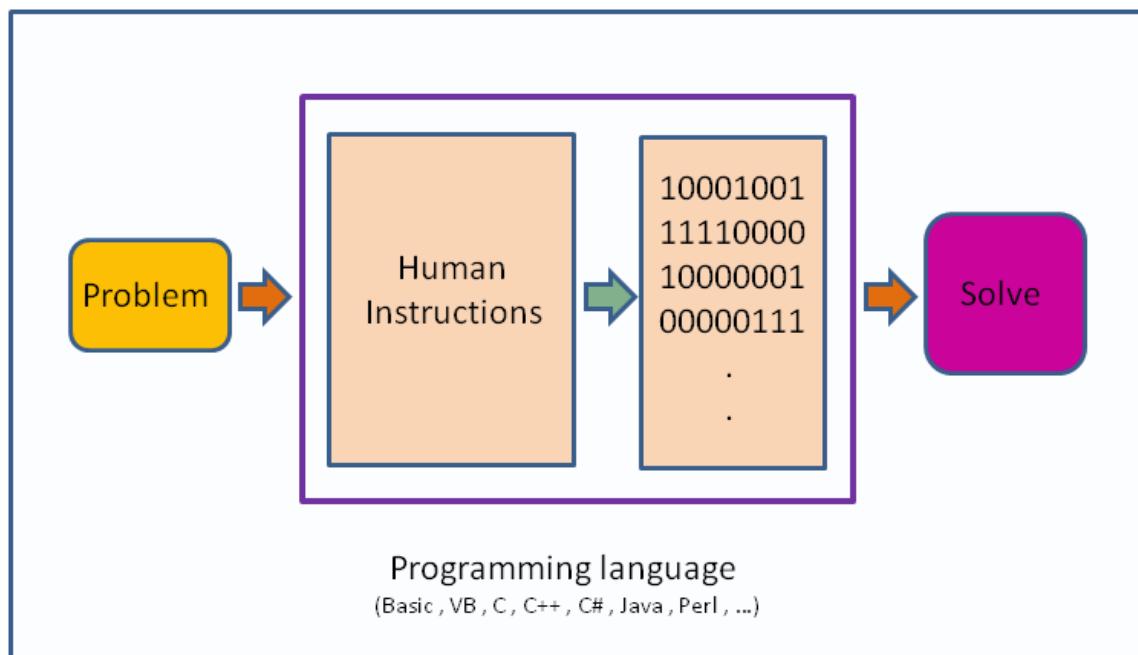
Om programmeren mogelijk te maken en instructies te kunnen geven moeten wij met de computers kunnen communiceren. Computers kunnen alleen de machinecode verstaan, geen talen, zoals Engels of Nederlands. Om deze communicatie mogelijk te maken, bestaan programmeertalen, uitgelicht in [\*\*3.2 Programmeertalen\*\*](#). Met behulp van de [\*\*3.1.4 IDE's\*\*](#) en [\*\*3.1.5 Compilers\*\*](#) kunnen wij sneller en efficiënter programmeren.

### 3.1.1 Programmeertalen

Een programmeertaal is een vocabulaire en, door mensen, leesbare taal die wordt gebruikt voor het maken van desktop applicaties, mobiele applicaties, websites, games. Verder wordt het gebruikt om veel meer andere ontwikkelingen simpeler of zelfs mogelijk te maken.

Elke programmeertaal heeft unieke trefwoorden (woorden die begrepen kunnen worden), een speciale syntax voor het organiseren van de instructies en een paar eigenschappen die de taal voor bepaalde doelen geschikt maken.

De broncode van een programma bestaat uit twee delen: Allereerst de human readable en high level instructies, die zijn geschreven in deze taal. En dezelfde instructies, maar dan geschreven in een andere taal, die in het geval van [Compiled language](#), [Machine code](#) wordt genoemd, en in het geval van geïnterpreteerde talen, [Bytecode](#) wordt genoemd. Machine code wordt door de compiler gecompileerd en de byte code wordt door de [Interpreter](#) geïnterpreteerd.



---

### **3.1.2 Het nut van programmeertalen**

Op basis hiervan werken zowel de moderne computers als de computers van 100 jaar geleden, alleen met binaire getallen. Een bit is de kleinste gegevenseenheid die er is, die slechts een waarde van 1 of 0 kan hebben. Deze enen en nullen staan voor de elektrische toestand en het stroomsingaal, dat aan of uit is in de elektronica. Computers zijn in staat om, met behulp van de binaire code, getallen in de vorm van digitale enen en nullen weer te geven. Deze worden in de CPU, RAM en de Storage opgeslagen.

Omdat bits zo klein zijn, worden ze bij programmeren heel weinig gebruikt. Daarom worden ze meestal samengevoegd tot een groep van acht om een Byte te vormen. Een Byte kan een waarde hebben van 0 tot 255 ( $2^8 = 256$ ), dit is net genoeg ruimte om een ASCII character op te kunnen slaan, zoals 'h'. Computers kunnen ook getallen die groter zijn dan 255 opslaan. Hiervoor worden er een paar Bytes samengesteld in een variabele, zodat er meer dan 256 combinaties van nullen en enen mogelijk zijn. Moderne computers, die 64-bit zijn (ofwel 8 bytes), kunnen dus een getal met een maximale waarde van 18,446,744,073,709,551,616 ( $2^{64}$ ) in een variabele opslaan. Voor oudere computers, die 32-bit waren, is deze waarde maar 4,294,967,296 ( $2^{32}$ ).

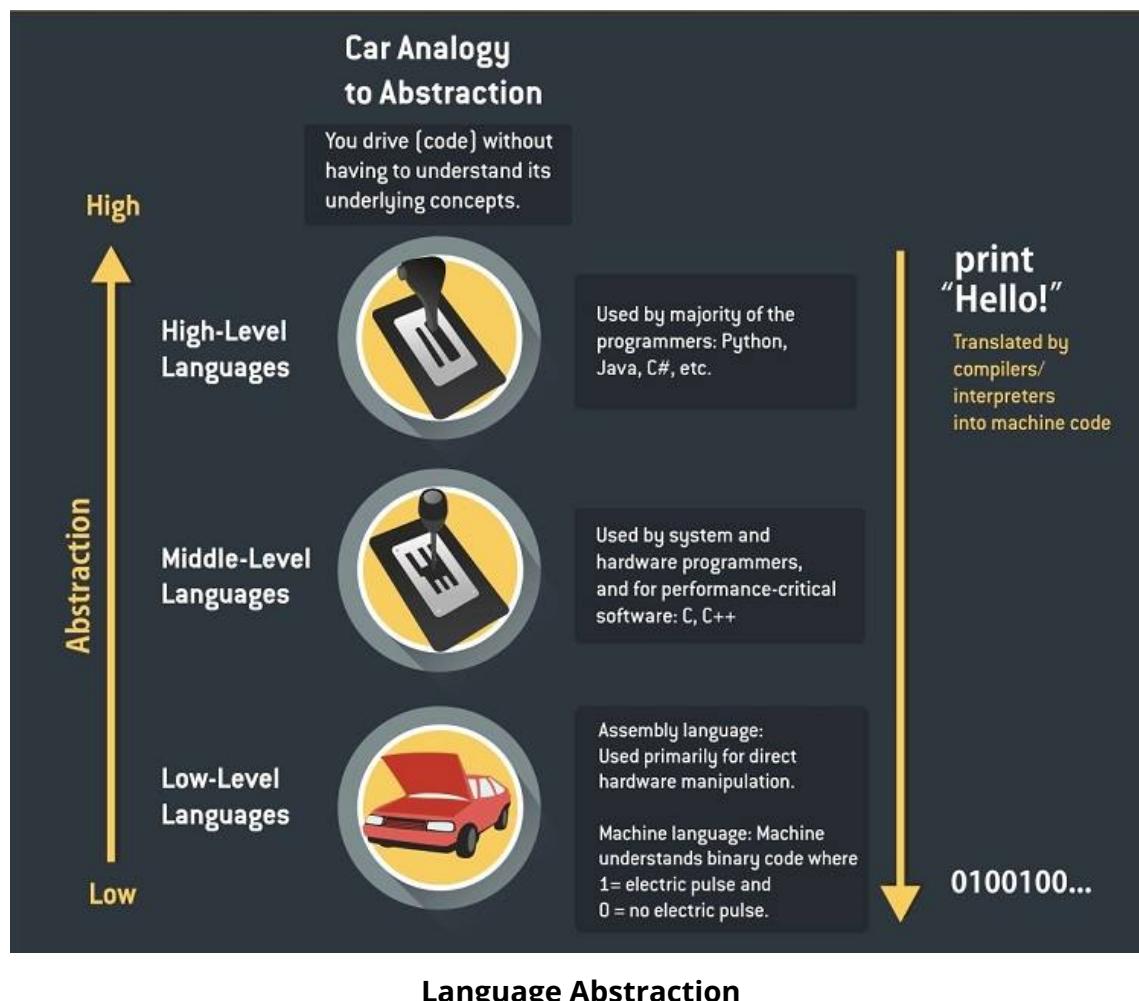
Er zijn meer dan miljoenen, of zelfs miljarden, bytes nodig voor het maken van een computerprogramma. Het is onmogelijk voor de mens om deze reeks van bytes zelf te typen of te bedenken. Het programmeren in bytes is vrijwel onmogelijk, omdat het zo abstract en tijdrovend is. Daarnaast wordt de code onleesbaar en moeilijk te onderhouden. Hierdoor zou niemand kunnen begrijpen wat een programma doet door alleen naar de bytes te kijken. Alleen ontwikkelaars die de performance belangrijk vinden, maken de keuze om in bytecode te programmeren. Het aantal tussenstappen in een programma is direct gerelateerd aan de performance van het programma. Als er minder tussenstappen tussen het programma en de machinecode zitten, werkt de code veel sneller en efficiënter.

Elke programmeertaal heeft voordelen en nadelen. Daarnaast zijn bepaalde talen beter geschikt voor bepaalde soorten taken. Er is dus geen beste taal.

### 3.1.3 Soorten programmeertalen

Een programmeertaal maakt het programmeren simpeler. Het is namelijk minder abstract dan de machinecode en bytecode en het wel kan worden gelezen door de mens. Elk level van een programmeertaal haalt steeds meer abstracte aspecten van het programmeren weg. Op basis van de abstractieniveaus worden programmeertalen in drie categorieën verdeeld. Elk niveau is kenmerkend voor de eigenschappen van een programmeertaal.

- **Low Level programmeertalen**
- **Medium Level programmeertalen**
- **High Level programmeertalen**



## Low Level programmeertalen

Een low level programmeertaal is een taal die geen of maar één abstractieniveau van hardware heeft. Er zijn twee low level programmeertalen, de Machine taal en de Assembly.

- **Machinetaal:** Machine taal is de oorspronkelijke programmeertaal van een cpu. Doordat de computers alleen binaire getallen kunnen verstaan, moeten de instructies en de broncode uiteindelijk worden vertaald naar binaire getallen. Aangezien machine taal geen abstractie heeft, is programmeren in deze taal een uitdaging. Omdat het niet leesbaar is, is machinetaal ook gevoelig voor fouten en vereist het veel onderhoud. Verschillende cpu's hebben verschillende machines codes en een verschillende [Instruction set architecture](#). Machine taal is dus niet overdraagbaar naar een ander platform.
- **Assembly:** Sommige commando's in deze talen zijn wel leesbaar voor mensen, zoals verplaatsen, toevoegen, sub, enzovoort. Deze talen worden in het Engels geschreven en zijn daarom veel simpeler en makkelijker dan machinetaal. Aangezien computers instructies alleen in machine taal kunnen lezen, is een vertaler nodig die de assembleertaal omzet naar machinetaal. Zo'n vertaler heet een [Assembly language](#), die wordt gebruikt om de code te vertalen. Omdat assemble code hoger in de hiërarchie staat dan machinecode, is deze langzamer.

- Assembly languages are also low-level, but more abstracted
- They use simple **mnemonics**
- They are also specific to hardware and have a \*mostly\* **one-to-one** relationship with machine code

**operator operand**

*Instruction*

**ADD R2, 10**

*Assembler*

**00000001 00101010**

**Assembly Language**

## Medium Level programmeertalen

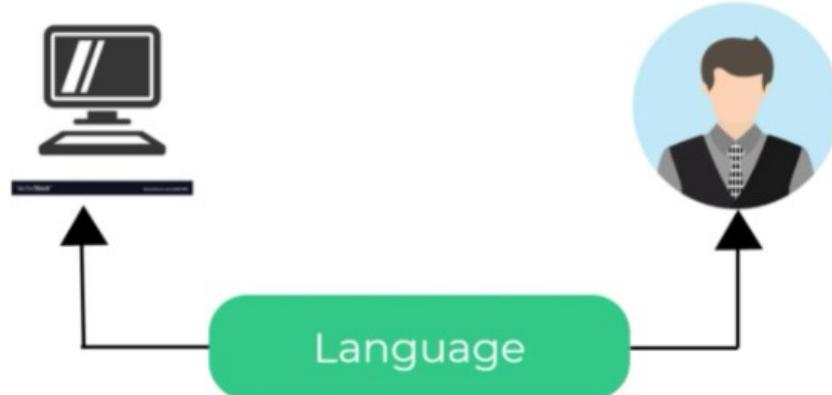
Programmeertalen die kenmerken van zowel low level als high level programmeertalen hebben worden medium level genoemd. Het voordeel van dit soort talen is dat de presentatie en efficiëntie veel hoger is dan high level languages en dat ze ook veel simpeler te onderhouden zijn dan low level languages.

C, C++<sup>(1)</sup> en Java zijn voorbeelden van Medium Level talen. C wordt beschouwd als een medium level taal, omdat deze taal slechts 32 trefwoorden heeft.

Machinetaal heeft geen abstractie, assembly talen hebben weinig abstractie en medium level talen hebben iets meer abstractie. Daardoor worden medium level talen veel gebruikt wanneer de presentatie belangrijker is dan de ontwikkelingskosten.

## Middle Level Language

Machine (computer)    Human(Programmer)



A language which is somehow closer to machine as well as human language

<sup>1</sup> Omdat C en C++ geen automatisch geheugenbeheer hebben, worden ze in de categorie van low programmeertalen geplaatst.

## High Level programmeertalen

High level programmeertalen zijn talen die hoger liggen in de stapel van lagen en veel abstracter zijn dan low level talen. Deze talen zitten vrij dicht bij de menselijke taal en zijn daardoor veel makkelijker om mee te programmeren, maar ook veel langzamer doordat ze veel abstracter zijn. High level talen worden met behulp van de interpreter vertaald naar de bytecode, op hetzelfde moment dat het programma wordt uitgevoerd. Doordat high level talen veel abstractie hebben, zijn ze makkelijker te leren, programmeren en debuggen.

Omdat moderne computers vrij snel zijn, compenseren de voordelen van high level talen de nadelen. High Level talen worden dan ook het meest gebruikt. Voorbeelden van High Level programmeertalen zijn Python, JavaScript, TypeScript, C#, Visual Basic, Ruby en veel meer.

### High-Level

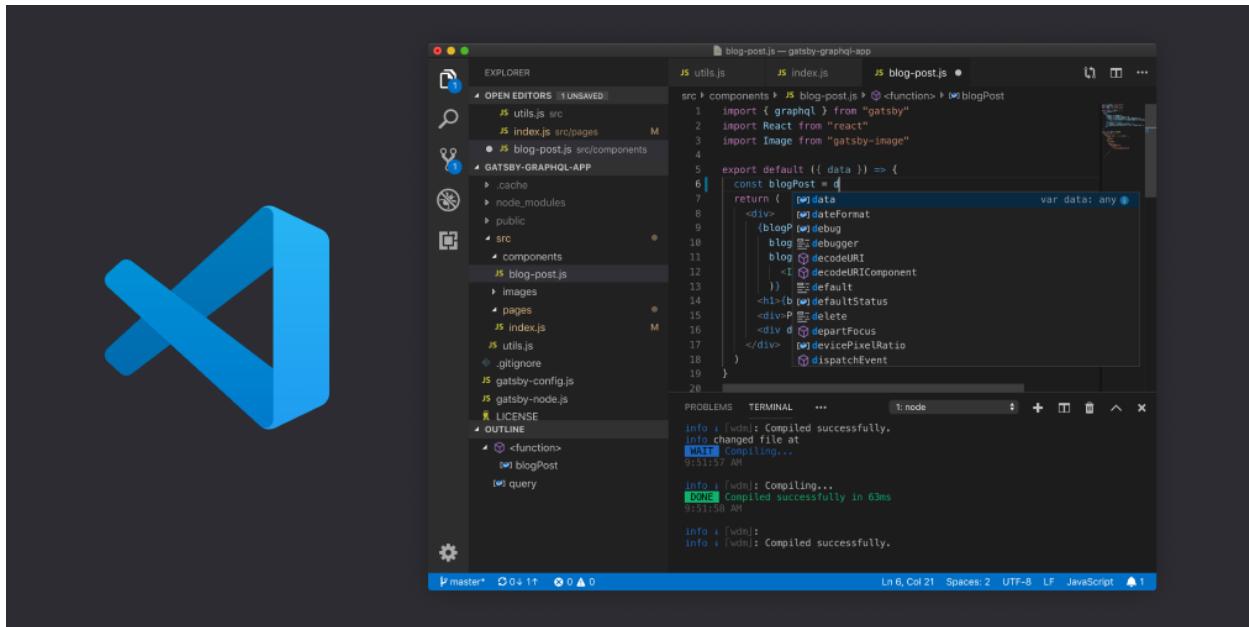


- Most like ordinary languages, so are far easier to use (still strict syntax)
- They are portable, meaning they can be executed on many computers i.e. you don't need to write hardware specific code, like low-level
- However, some high-level languages are more abstract than others...
- Less abstract: *pointers, strong types, enums, stacks, templates etc...*
- More abstract languages leave more work to be done at runtime, so are much slower to execute. With less abstract, you can optimise more

### High Level Languages

### 3.1.4 IDE's

Een IDE, [Integrated development environment](#), is een programmeeromgeving die de ontwikkelaars helpt met verschillende aspecten van het programmeren. Een van de belangrijkste voordelen van IDE's is dat ze krachtige toolset en handige hulpmiddelen hebben om programmeren eenvoudiger te maken.



Visual Studio Code

- **Snellere insteltijd:** Met behulp van een IDE hebben de ontwikkelaars toegang tot alle nodige tool sets, zonder van software te hoeven wisselen. Zonder een IDE moeten ontwikkelaars tijd besteden aan het configureren en klaarmaken van meerdere ontwikkelde tool sets, waardoor het veel tijd en geld kost.
- **Snellere en bug-vrije ontwikkeling:** Ontwikkelaars kunnen met behulp van de uitgebreide toolsets die IDE's hebben, een grotere productiviteit bereiken. Bijvoorbeeld, het melden van syntaxfouten, of andere gevaarlijke programmeurfouten tijdens programmeren. IDE's hebben ook toolsets die de ontwikkelaars helpen met het organiseren van de code, het voorkomen van fouten en veel meer.

---

Hieronder staan de belangrijkste toolsets die IDE's hebben om het programmeren eenvoudiger en meer tijd efficiënt maken:

- **Source-code editor**

Het schrijven van de code en het organiseren van elk bestand is een belangrijk onderdeel van het programmeren. IDE's maken dit proces makkelijker met verschillende toolsets zoals A.I. assisted [Auto complete](#), [Debugging](#), Syntax Correction en nog veel meer.

- **Syntax highlighting**

Trefwoorden zoals include, namespaces, classes, functies, variabelen, namen en getallen worden met verschillende kleuren aangetoond, zodat de code meer leesbaar wordt. Hierdoor zullen er minder fouten en bugs ontstaan en het ontwikkelen zal minder tijd kosten.

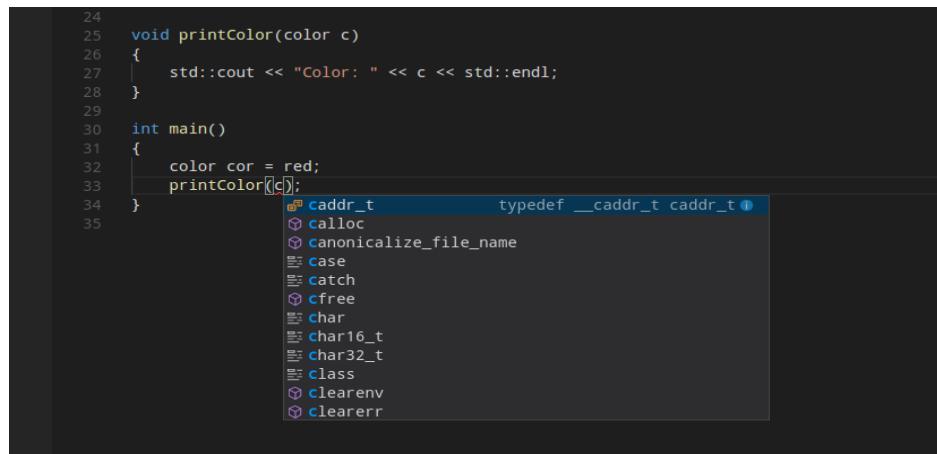
```
class myFile
{
public:
    myFile(char* s){};
};

int main(int argc, char* argv[])
{
    myInput = myFile(argv[1]);
    myOutput = myFile(argv[2]);
}
```

Syntax Highlighting

- Auto complete

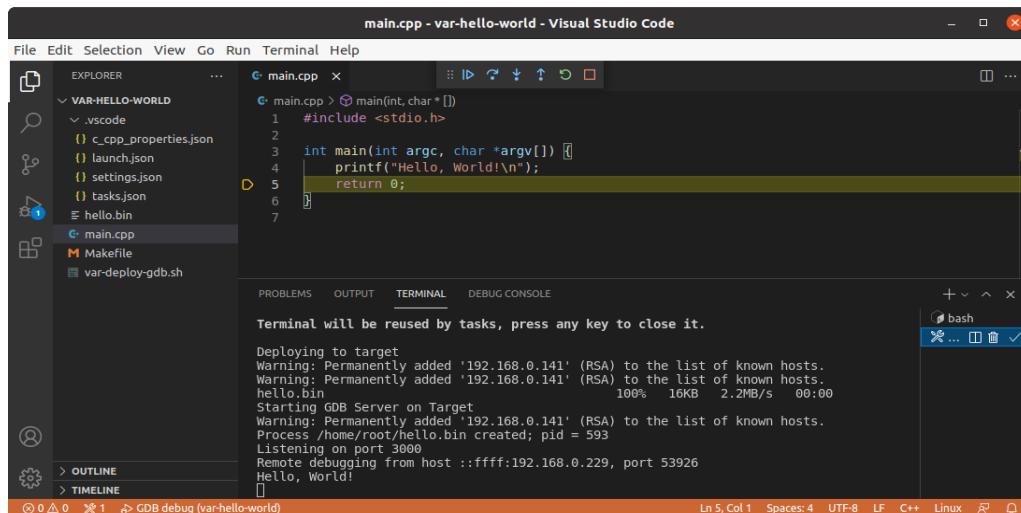
Gebaseerd op wat er al is getypt, kunnen IDE's gokken wat de programmeur zal gaan typen. In de onderstaande afbeelding is te zien dat, nadat de functie "printColor" is getypt, "printColor(**cor**);" automatisch wordt aangetoond. Om het toe te voegen, hoeft de ontwikkelaar alleen maar op enter of tab te klikken of er op klikken met de muis.



Auto Completion

- Debuggen

Elke ontwikkelaar maakt fouten en schrijft het programma met bugs. IDE's hebben debugging tool sets, waarmee verschillende functies en waarden van variabelen op het moment zelf en lijn voor lijn geïnspecteerd kunnen worden. Er zijn ook toolsets die hints geven over de gemaakte fouten.



Debugging

### 3.1.5 Compilers

Het vertalen van de geschreven high level code naar machinecode en CPU instructies wordt door de [Compiler](#) gedaan.

De broncode wordt in een high level en human readable taal, zoals python of JavaScript, geschreven. De ontwikkelaar schrijft de broncode met behulp van een code editor of een Integrated Development Environment (IDE), waarbij het in een of meer tekstbestanden wordt opgeslagen. Een compiler, die de geprogrammeerde taal ondersteunt, leest de bestanden, analyseert de code en vertaalt deze naar de machinecode die geoptimaliseerd en geschikt is voor het doelplatform.

Elk CPU- en besturingssysteem heeft een unieke Instruction set architecture. Elke compiler is speciaal voor een bepaald platform gemaakt. Het resulterende bestand van de compiler wordt soms ook wel objectcode genoemd (wat niet gerelateerd is aan objectgeoriënteerd programmeren). Dit bestand bestaat volledig uit binaire getallen (enen en nullen), zodat het gelezen en vervolgens uitgevoerd kan worden door de CPU.

Een compiler kan bijvoorbeeld een machinecode maken voor het Linux x64-platform, het Linux ARM 64-bits platform of Windows. De conventie is dat de ontwikkelaar ervoor moet zorgen dat zijn programma compatibel is met elk platform.



Van broncode naar Machinecode

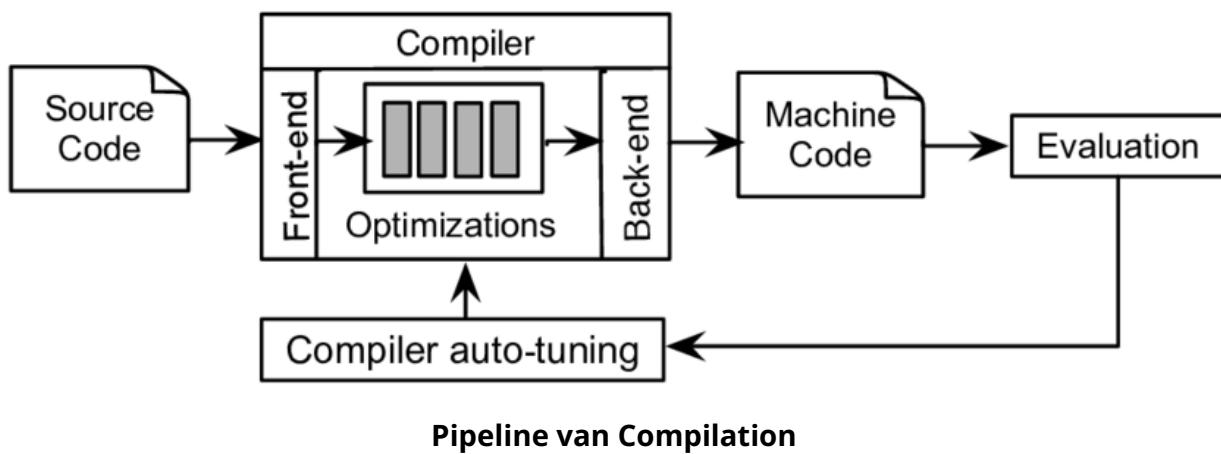
## Optimalisatie

Het andere belangrijke voordeel van de compiler is het optimaliseren van de broncode tijdens het compileren. De compiler probeert het programma efficiënter en sneller te maken door de broncode, met behulp van de geïntegreerde algoritmen, te scannen en optimaliseren.

Tijdens het compilatieproces worden de instructies op hoog niveau vervangen door zeer efficiënte instructies op laag niveau. Voordat deze stap is voltooid, grijpt de compiler in en verbetert de code automatisch.

Het optimalisatieproces van de code moet aan drie regels voldoen:

- Het resultaat mag op geen enkele manier de inhoud en de functionaliteit van het programma veranderen.
- Optimalisatie zou de snelheid van het programma moeten verhogen en, indien mogelijk, zou het programma minder resources moeten gebruiken.
- Optimalisatie moet zelf snel zijn en mag het algehele compilatieproces niet vertragen.



---

Hierbij geven wij een simpele voorbeeld van optimalisatie door een compiler:

```
int main()
{
    item = 10;
    do
    {
        item += 10;
    } while (item < 100);

    return 0;
}
```

In de bovenstaande code neemt de waarde van de variabele 'item' in elke iteratie van de while-loop toe met 10, zodat de variabele uiteindelijk 100 wordt. Een goed ingestelde compiler zal de code als volgt optimaliseren:

```
int main()
{
    item = 100;

    return 0;
}
```

Het programma van de geoptimaliseerde code zou minder werk en CPU clock cycles vereisen. Dit komt doordat de compiler ziet dat de variable item uiteindelijk een waarde van '100' moet hebben en zou dan automatisch 'item = 10' met 'item = 100' vervangen. Op deze manier gaat de snelheid in het algemeen omhoog en het resourcegebruik omlaag.

## Soorten bugs

In een programmeertaal moet elke zin volledig duidelijk zijn. Daardoor zijn er heel precieze grammaticaregels voor alle onderdelen van de taal. Een software bug ontstaat wanneer de code helemaal niet gecompileerd kan worden, wanneer het een onjuist of onverwacht resultaat oplevert, of wanneer het op onbedoelde manieren functioneert.

In totaal zijn er 5 verschillende complicatie errors in C:

- **Syntax Error**

Fouten die optreden als de compiler de geschreven code niet kan begrijpen. Zoals spel- en interpunctiefouten, onjuiste trefwoorden, enzovoort, waardoor er een foutmelding wordt getoond door de compiler.

- **Run Time Error**

Fouten die tijdens de uitvoering van het programma optreden. Deze fouten treden dus op nadat het programma succesvol is gecompileerd. Als het programma een ongeldige functie tegenkomt, ontstaat er een runtime-error.

- **Logical Error**

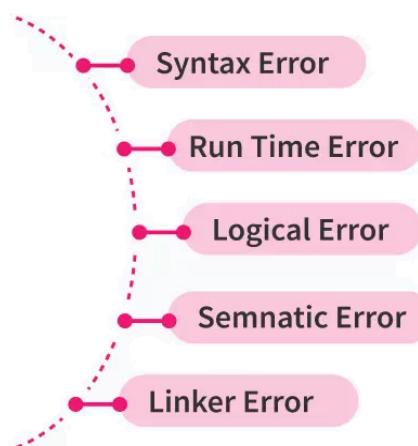
Soms krijgen wij een resultaat dat wij niet verwachten, hoewel de code foutloos blijkt te zijn. In dit geval is er een logische fout opgetreden.

- **Semantic Error**

Fouten ontstaan als de code niet logisch is voor de compiler, ook al is deze syntactisch correct. Het gaat hier over de betekenis van de woorden.

- **Linker Error**

Linker is een programma dat de objectbestanden van de compiler tot een enkel uitvoerbaar bestand combineert. Linker fouten treden op wanneer het uitvoerbare bestand niet gegenereerd kan worden, ook als de code succesvol is gecompileerd.



### 3.1.6 ASCII

ASCII (American Standard Code for Information Interchange) is het meest gebruikte character encoding format voor het opslaan van teksten en gegevens in computers. Alle gegevens in websites, applicaties, programma's, sms, string variabelen en etc worden automatisch met behulp van de ASCII tabel omgezet naar de equivalente waarde en vervolgens opgeslagen. De ASCII is 1 byte groot (8 bits) en heeft genoeg ruimte voor een karakter. Dus ASCII kan een maximale waarde hebben van 255 ( $2^8 - 1$ ). De standaard versie van ASCII tabel begint bij 0 en eindigt bij 127. De uitgebreide ASCII tabel begint ook bij nul, maar ondersteunt meer karakters en eindigt bij 255.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	Ø	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	Ø	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	:	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Dec	Hex	Char									
128	80	ç	160	A0	á	192	C0	ł	224	E0	α
129	81	ü	161	A1	í	193	C1	ł	225	E1	ß
130	82	é	162	A2	ó	194	C2	ł	226	E2	Γ
131	83	â	163	A3	ú	195	C3	ł	227	E3	π
132	84	ä	164	A4	ñ	196	C4	—	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	+	229	E5	σ
134	86	å	166	A6	²	198	C6	ƒ	230	E6	μ
135	87	ç	167	A7	°	199	C7	॥	231	E7	τ
136	88	ê	168	A8	¸	200	C8	Ł	232	E8	Φ
137	89	ë	169	A9	¬	201	C9	Ƒ	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	܂	234	EA	܂
139	8B	í	171	AB	܂	203	CB	܂	235	EB	܂
140	8C	î	172	AC	܂	204	CC	܂	236	EC	܂
141	8D	ì	173	AD	܂	205	CD	=	237	ED	܂
142	8E	Ä	174	AE	«	206	CE	܂	238	EE	܂
143	8F	Å	175	AF	»	207	CF	܂	239	EF	܂
144	90	É	176	B0	܂	208	D0	܂	240	F0	܂
145	91	æ	177	B1	܂	209	D1	܂	241	F1	܂
146	92	Æ	178	B2	܂	210	D2	܂	242	F2	܂
147	93	ð	179	B3	܂	211	D3	܂	243	F3	܂
148	94	ö	180	B4	܂	212	D4	܂	244	F4	܂
149	95	ò	181	B5	܂	213	D5	܂	245	F5	܂
150	96	û	182	B6	܂	214	D6	܂	246	F6	܂
151	97	ù	183	B7	܂	215	D7	܂	247	F7	܂
152	98	ÿ	184	B8	܂	216	D8	܂	248	F8	܂
153	99	Ö	185	B9	܂	217	D9	܂	249	F9	܂
154	9A	Ü	186	BA	܂	218	DA	܂	250	FA	܂
155	9B	¢	187	BB	܂	219	DB	܂	251	FB	܂
156	9C	£	188	BC	܂	220	DC	܂	252	FC	܂
157	9D	¥	189	BD	܂	221	DD	܂	253	FD	܂
158	9E	€	190	BE	܂	222	DE	܂	254	FE	܂
159	9F	f	191	BF	܂	223	DF	܂	255	FF	܂

Het opslaan van '**Sadra**' in een string variabele vereist 6 bytes geheugen. Elk karakter van de variabele wordt in een byte opgeslagen. De zesde byte is voor de Null terminating character ('/0' of '0'), die het einde van de string aangeeft. Dit komt doordat de compiler niet weet hoe groot een string variabel is. Het gaat dus door totdat het een '0' tegen komt.

"Sadra" → 83 97 100 114 97 0  
 'S' 'a' 'd' 'r' 'a' '/0'

---

## 3.2 Programmeertalen

In dit hoofdstuk geven wij een korte beschrijving van alle talen die wij in ons project gebruiken en bespreken we de voordelen en nadelen van elke taal.

### 3.2.1 C, C++

C en C++ zijn talen die voornamelijk in het programmeren van operating systems en kernels, game engines en simulatie software worden gebruikt, waar efficiëntie en snelheid een hoge prioriteit hebben. Dit komt doordat C en C++ lage-level API's hebben die directe controle op de hardware mogelijk maken. Deze talen worden ook wel de moeder van alle talen genoemd. Bijna alle andere talen zijn gebaseerd op de basisconcepten van C. Deze talen hebben geen [Automatische memory management](#) systemen geïntegreerd zoals [Garbage collection](#), (deze zijn de verantwoordelijkheid van de ontwikkelaar), daardoor hebben ze veel minder runtime overhead dan andere high level talen, zoals python en JavaScript. Dit zorgt ervoor dat de performance heel hoog blijft.

C++ is een uitgebreidere versie van C die [Object oriented programming](#) ondersteunt. C++ heeft veel helper classes geïntegreerd zoals de Standard Library (STD). Dit maakt de taal meer compleet en eenvoudiger, en daardoor wordt het meer dan C gebruikt.

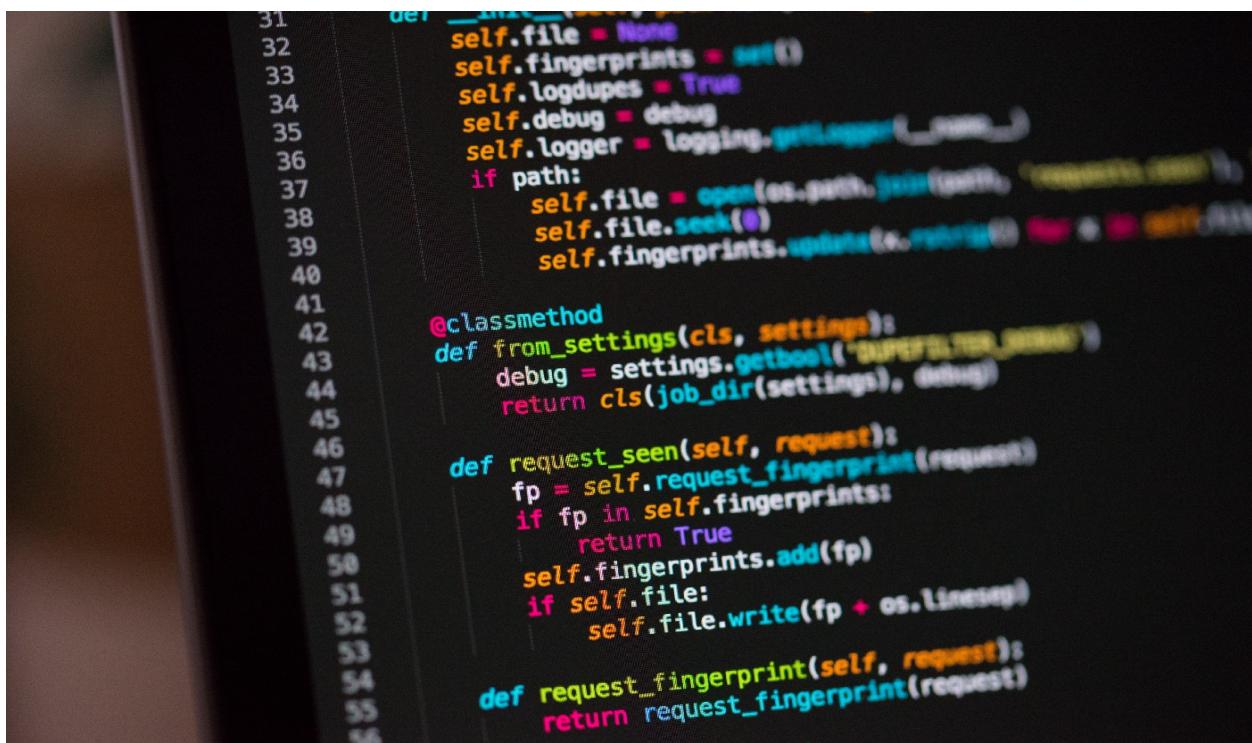
Hieronder staan de belangrijkste voordelen van C en C++:

- Objectgeoriënteerd programmeren met classes, inheritance etc (In C++).
- C kan in bijna alle computers, IoT en embedded systemen gebruikt worden, doordat het low level en hardware compatibel is.
- De ontwikkelaar heeft veel mogelijkheden voor memory management.
- Geen runtime overhead vanwege de afwezigheid van een automatische memory management systeem en het feit dat het een [Dynamic typed language](#) is.
- Snel en efficiënt, doordat C low level is en de code veel geoptimaliseerd kan worden door de compiler.

### 3.2.2 Python

Python is een [geïnterpreteerd](#), [objectgeoriënteerd](#), high level programmeertaal met dynamische semantiek. De ingebouwde high level datastructuren, gecombineerd met dynamische types en dynamisch bindingen maakt python een goede keuze voor het ontwikkelen van websites en software, automatisering in de industrie, zelfrijdende auto's, analyse, visualisatie, simulatie van data, alledaagse apparaten zoals IoT, smart home applicaties, enzovoort. Omdat python eenvoudig te leren is en niet veel ontwikkelingstijd vereist, wordt het ook door niet programmeurs zoals accountants en financiënen gebruikt.

Om de ontwikkeling van artificial intelligence en neuraal netwerk simpeler te maken, wordt python heel vaak samen met de [TensorFlow](#) library gebruikt.



```
31     def __init__(self, path=None, debug=False):
32         self.file = None
33         self.fingerprints = set()
34         self.logdups = True
35         self.debug = debug
36         self.logger = logging.getLogger(__name__)
37         if path:
38             self.file = open(os.path.join(path, 'fingerprint.log'), 'a')
39             self.file.seek(0)
40             self.fingerprints.update(self._read())
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getbool('supervisor.debug')
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
54
55     def request_fingerprint(self, request):
56         return request_fingerprint(request)
```

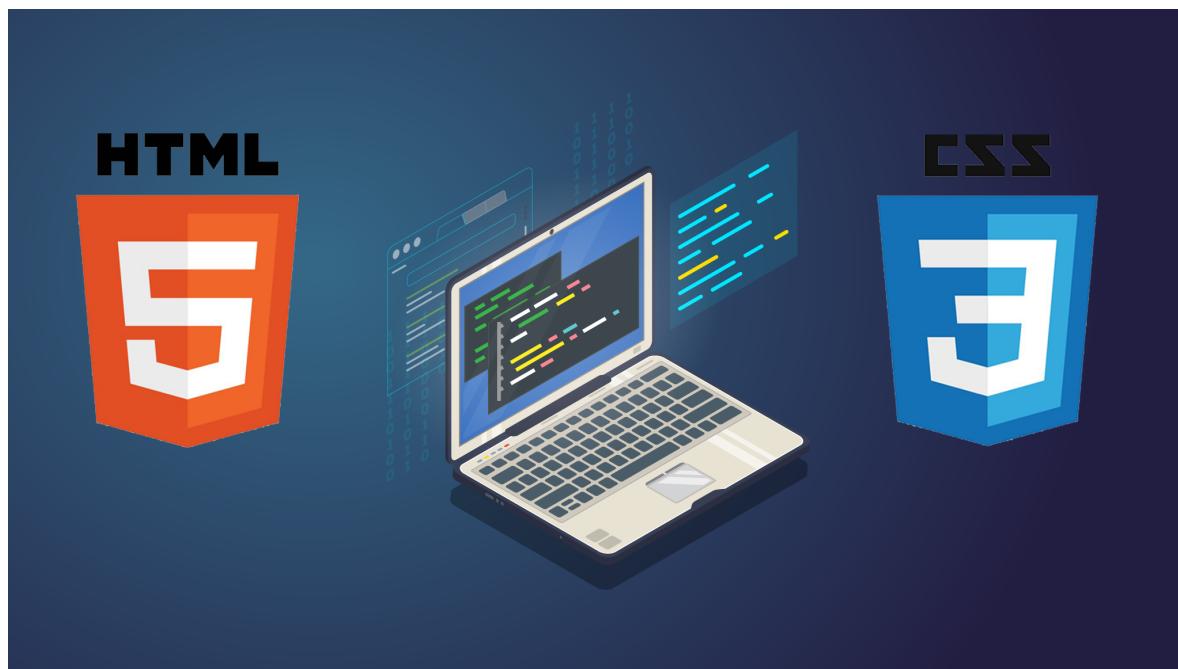
Python

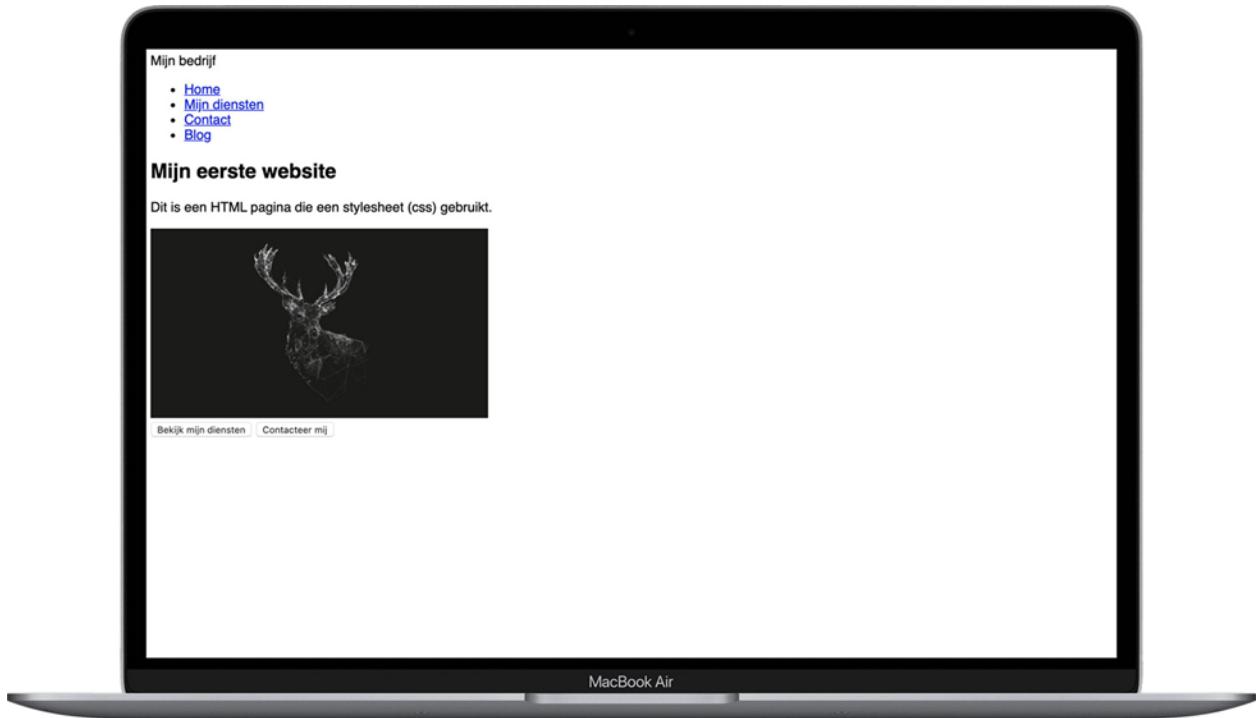
### 3.2.3 HTML, CSS en JavaScript

**HTML**, **Hyper Text Markup Language**, is een taal voor het maken van een webpagina. HTML is meer een opmaaktaal dan een programmeertaal. HTML bestanden worden door de webbrowser gelezen en alle elementen worden vervolgens in een webpagina weergegeven. Hyper Text houdt in dat alle bestanden door hyperlinks zijn verbonden. Als je op een link klikt dan kom je uit op een andere webpagina. HTML wordt dus als de structuur en basis van het web beschouwd.

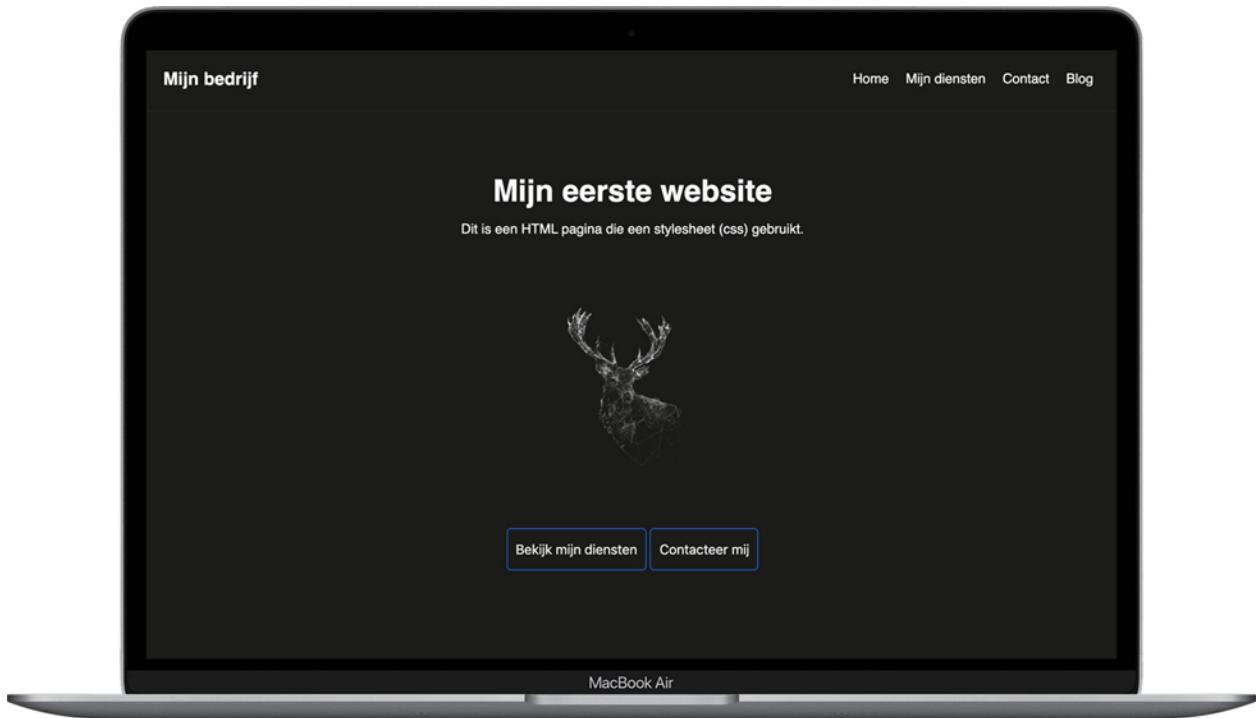
**CSS**, **Cascading Style Sheets**, beschrijft hoe de elementen van een HTML bestand gerenderd en weergegeven moeten worden op het scherm. Je kunt hierbij denken aan de kleur, waar elk element plaatsneemt, tekstgrootte etc. Voor HTML bestanden zonder een CSS styling, past de browser automatisch een standaardstijl toe, die de **User agent stylesheet** wordt genoemd. Meestal lijkt dit op een heel oude website zonder enige vorm. (figuur 1) Met CSS kun je dus het uiterlijk van een HTML website bepalen.

**Javascript** is een high level scripttaal, die voornamelijk bij web ontwikkelingen wordt gebruikt om webpagina's functies toe te voegen en ze interactiever te maken. Javascript is, in tegenstelling tot CSS en HTML, een programmeertaal met meer mogelijkheden.





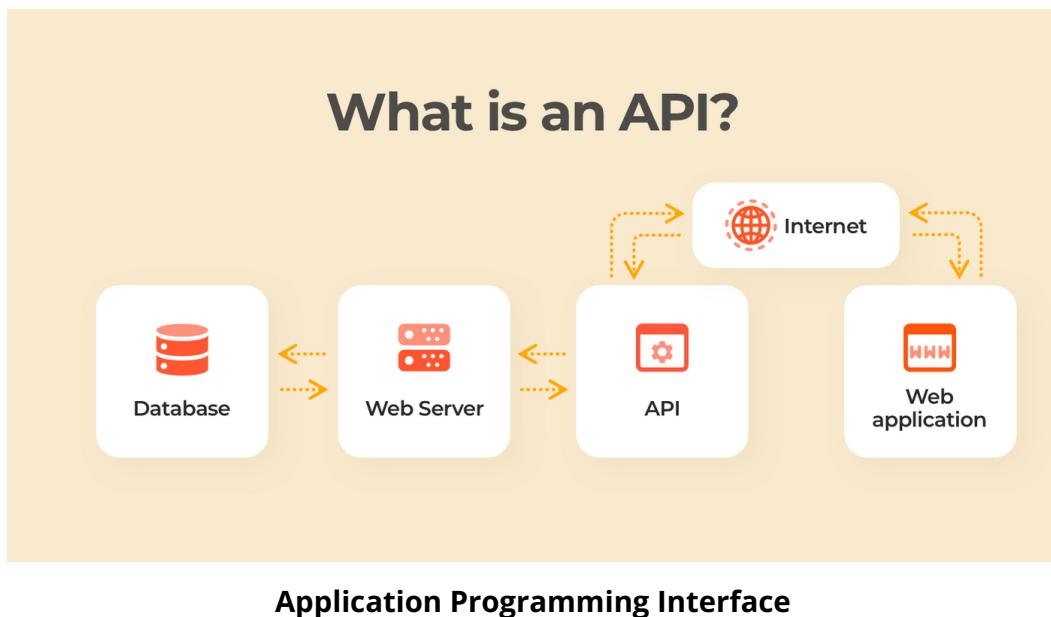
Figuur 1 - HTML pagina zonder CSS



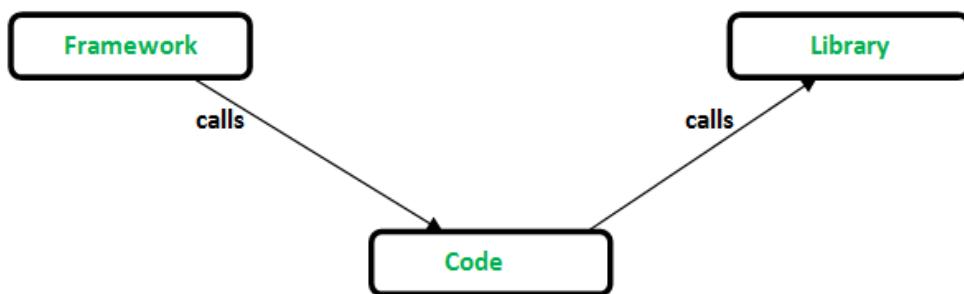
Figuur 2 - HTML pagina met CSS

### 3.2.4 API en Libraries

API of [Application programming interface](#), is een verzameling van functies, classen en helper modules, waarmee twee of meerdere programma's met elkaar kunnen communiceren. API's vormen de grens tussen verschillende lagen van abstractie, zodat het minder abstracte werk van een high level programma door een ander programma uitgevoerd kan worden. API is dus te vergelijken met een taal waarmee mensen elkaar makkelijker kunnen begrijpen. Elke taal heeft zijn specifieke en unieke regels en moeilijkheidsgraad.



[Libraries](#), bibliotheek in het Nederlands, is de broncode van een API. De bibliotheek verwijst naar de code zelf, terwijl de API naar de interface verwijst. Dus de functies, classen en modules die gepubliceerd worden en beschikbaar zijn voor de ontwikkelaar.



### 3.2.5 TensorFlow

Het programmeren van een neural network is niet makkelijk en de nodige ingewikkelde matrixberekeningen vereisen een hoge kennis van wiskunde. Daardoor worden er hulpprogramma's of libraries gebruikt die al die moeilijke berekeningen doen. Het maken van een eigen library zal veel tijd en moeite kosten, en de prestaties zullen veel lager zijn.

Een voorbeeld van een neural network library is het populaire TensorFlow, met API's verkrijgbaar in Python, JavaScript en C++. Met behulp van TensorFlow kunnen de ontwikkelaars [Data flow diagram](#) maken: structuren die beschrijven hoe de data door een grafiek of een reeks van processing nodes gaat. Elke node in de grafiek vertegenwoordigt een wiskundige bewerking en elke verbinding of rand tussen nodes is een [Multidimensional Data Array](#) of [Tensor](#).



The screenshot shows a code editor with an 'index.html' file open. The code uses the TensorFlow.js library to create a simple neural network model. It includes imports for TensorFlow.js, defines a prediction function, creates a sequential model, adds a dense layer, compiles it with mean squared error loss and SGD optimizer, fits the model with sample data, and finally makes a prediction. The TensorFlow.js logo is overlaid on the right side of the code editor window.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs/dist/tf.min.js"></script>
5   </head>
6   <body>
7     <p id="prediction">Prediction(100): </p>
8     <script>
9       async function run(){
10         const model=tf.sequential();
11         model.add(
12           tf.layers.dense([
13             {
14               units: 1,
15               inputShape: [1],
16             }
17           ]);
18         );
19         model.compile({
20           loss: 'meanSquaredError',
21           optimizer: 'sgd',
22           metrics: ['mse']
23         });
24
25         const xs = tf.tensor1d([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]);
26         const ys = tf.tensor1d([2, 5, 8, 12, 14, 18, 21, 23, 26, 29]);
27
28         await model.fit(xs, ys, {epochs:100});
29         document.getElementById('prediction').innerText+=
30         model.predict(tf.tensor1d([100])).dataSync();
31       }
32     </script>
33   </body>
34 </html>
```

TensorFlow

---

## 3.3 De Hardware

### 3.3.1 Raspberry Pi

Raspberry Pi wordt meestal gebruikt door mensen met een grote interesse voor computers en elektronica, die een makkelijke, snelle en efficiënte mini computer willen hebben. Lage ontwikkelingskosten, modulariteit en het feit dat het open source is, maakt Raspberry Pi zeer geschikt voor weersvoorspellingen, schoolprojecten, hardware ontwikkelingen, robotica, geautomatiseerde apparaten, circuits.

De Raspberry Pi 4 Model B is de nieuwste en snelste serie van Raspberry Pi mini computers, met een hoge performance CPU die op een frequentie van 1.8GHz draait. Er is een selectie van 1GB, 2GB, 4GB en 8GB geheugen beschikbaar die allemaal op een frequentie van 3200 MHz draaien. Om meer functionaliteiten toe te voegen en allerlei externe modules en sensoren aan te sluiten, zijn er 40 GPIO pins beschikbaar die aan de bovenkant makkelijk bereikbaar zijn.



**Raspberry Pi 4B**

Voor het scannen van de omgeving, uitrekenen van juiste richting, uitrekenen van de minimale afstand en het vervolgens vermijden van obstakels, moeten er moeilijke wiskundige formules worden opgelost. Hiervoor is een snelle computer nodig. De performance van de Raspberry Pi 4B maakt het mogelijk om een goede A.I. te ontwikkelen.

### 3.3.2 Espressif ESP32

ESP32 is een high performance en energiezuinige chip, ontwikkeld door Espressif Systems. Deze chip heeft een Tensilica Xtensa LX6-microprocessor dual core CPU met een kloksnelheid van 240 MHz, 512 Kilobytes SRAM, geïntegreerde hall effect en temperatuur sensor en heeft Wi-Fi (802.11b/g/n) en dual mode [Bluetooth classic](#) en [LE](#).<sup>(1)</sup> De WiFi ondersteunt de encryptiemethode WPA2 en kan ook als access point of sniffer werken. Via 30 of 36 GPIO pinnen (hangt af van het model) zijn er data porten zoals UART, I2C, SPI, DAC, ADC (12 bit) beschikbaar. Alle GPIO pinnen ondersteunen input en output modus.

ESP32 development boards worden in de onderstaande IoT apparaten veel gebruikt:

- Slimme industriële apparaten, zoals Programmable Logic Controllers (PLC's)
- Slimme medische apparatuur, zoals draagbare gezondheidsmonitoren
- Slimme energie apparaten, zoals HVAC systems en thermostaten
- Slimme beveiligings apparaten, zoals bewakingscamera's en slimme sloten



ESP32

<sup>1</sup> Niet in alle modellen

### 3.3.3 YDLidar X4

Lidar of [\*\*Light Detection And Ranging\*\*](#), is een technologie die, met behulp van laserpuls, de afstand tot een object of oppervlak bepaalt. De laser sensor zendt een laserpuls uit. Deze puls wordt daarna gereflecteerd. De afstand van de laser sensor tot het object wordt bepaald met behulp van de lichtsnelheid, door de verstreken tijd tussen het uitzenden van de laserpuls en het ontvangen van de reflectie te meten. Lidar technologie wordt in verschillende gebieden gebruikt, zoals zelfrijdende auto's, robots, smart cities, mapping van de omgeving, industriële verticals en IoT apparaten.

[\*\*YDLidar X4\*\*](#) lidar is een 360-graden tweedimensionaal laser range scanner of Lidar. Dit apparaat maakt gebruik van het **Driehoeksmeting principe<sup>(1)</sup>** om afstand te meten. Samen met een geschikt optisch, elektrisch en algoritmisch ontwerp kan het een zeer nauwkeurige afstandsmeting bereiken.

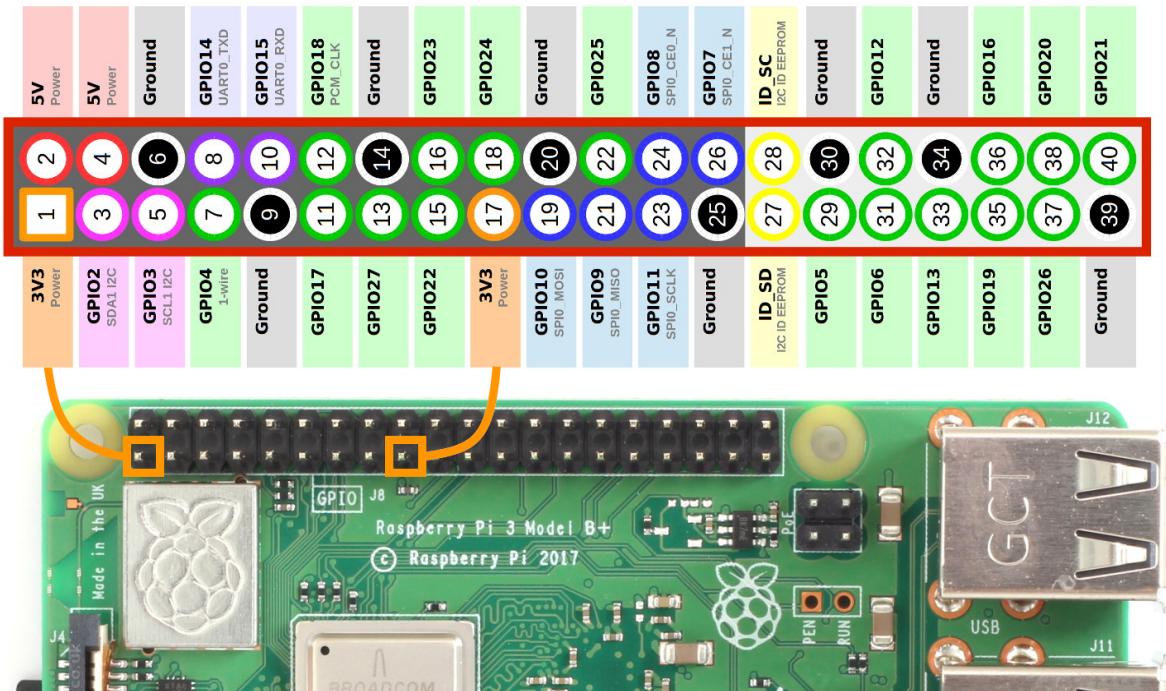


**YDLidar X4**

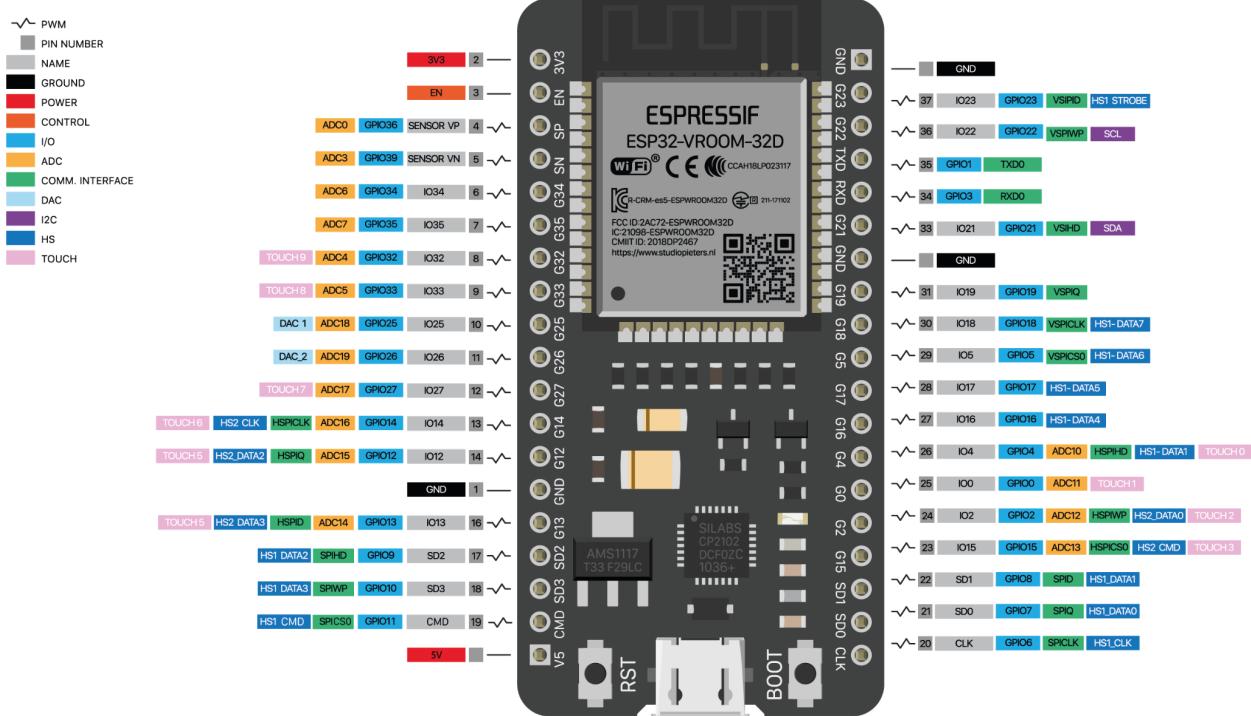
---

<sup>1</sup> [\*\*Driehoeksmeting\*\*](#) is een meetmethode die gebruikmaakt van het feit dat een driehoek volledig bepaald is als één zijde (de basis) en de aanliggende hoeken bekend zijn.

### 3.3.4 Pinouts, wirings



Raspberry Pi 3B+, 4B+ pinout



ESP32 pinout

## Polar coordinate system definition

For secondary development, X4 internally defines a polar coordinate system.

Pole: the center of the rotating core of the X4;

Positive direction: clockwise;

Zero angle: directly in front of the X4 motor;

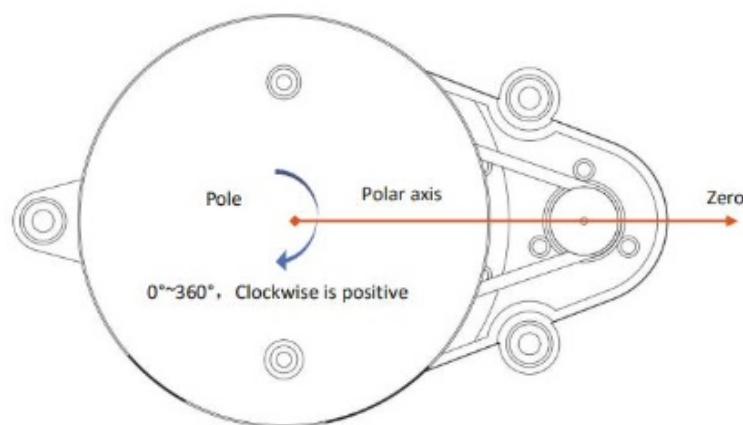
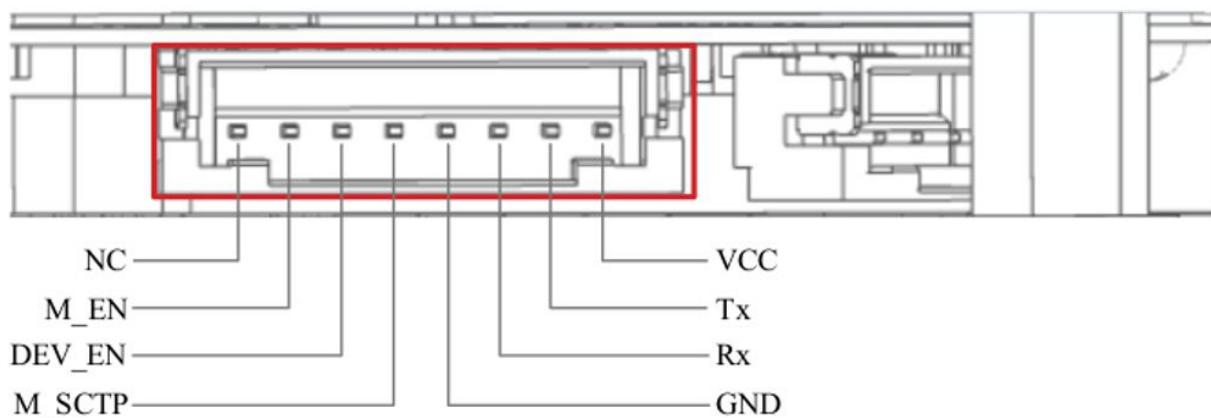


FIG4 YDLIDAR X4 POLAR COORDINATE SYSTEM DEFINITION



YDLidar X4 pinout



This voltage converter module can convert input voltages of 9 - 35 V down to an output voltage of 5 V.

MAIN FEATURES	
Model	Stepdown converter with fixed 5 V output

Dimensions	31 x 46 x 19 mm
Weight	17 g

Items delivered	Voltage converter module
-----------------	--------------------------

FURTHER SPECIFICATIONS	
Input voltage	9 - 35 V
Output voltage	5 V
Output current	5 A max.
Output power	25 W max.
Connector	3 pin screw terminal

FURTHER DETAILS	
Article No.	SBC-Buck02
EAN	4250236824017
Customs Tariff No.	85044030

### Joy-it SBC-Buck02

## Technical Specification

<b>Model</b>	SBC-MotoDriver2
<b>Driver</b>	L298N
<b>Logical voltage</b>	5V
<b>Drive Voltage</b>	5V – 35V
<b>Drive Current</b>	2A
<b>Power</b>	Max. 25W
<b>Dimensions (L x B x H)</b>	43mm x 43mm x 27mm
<b>Scope of delivery</b>	MotoDriver2
<b>EAN</b>	425023681513

### SBC-MotoDriver2

## 3.4 Artificial Intelligence

A.I. is een concept waarbij computers kunnen denken, leren en taken uitvoeren, waar menselijke intelligentie voor nodig is, met als uiteindelijk doel het oplossen van een probleem. A.I. is een heel algemeen concept en beschrijft veel verschillende algoritmen.



**Artificial Intelligence**

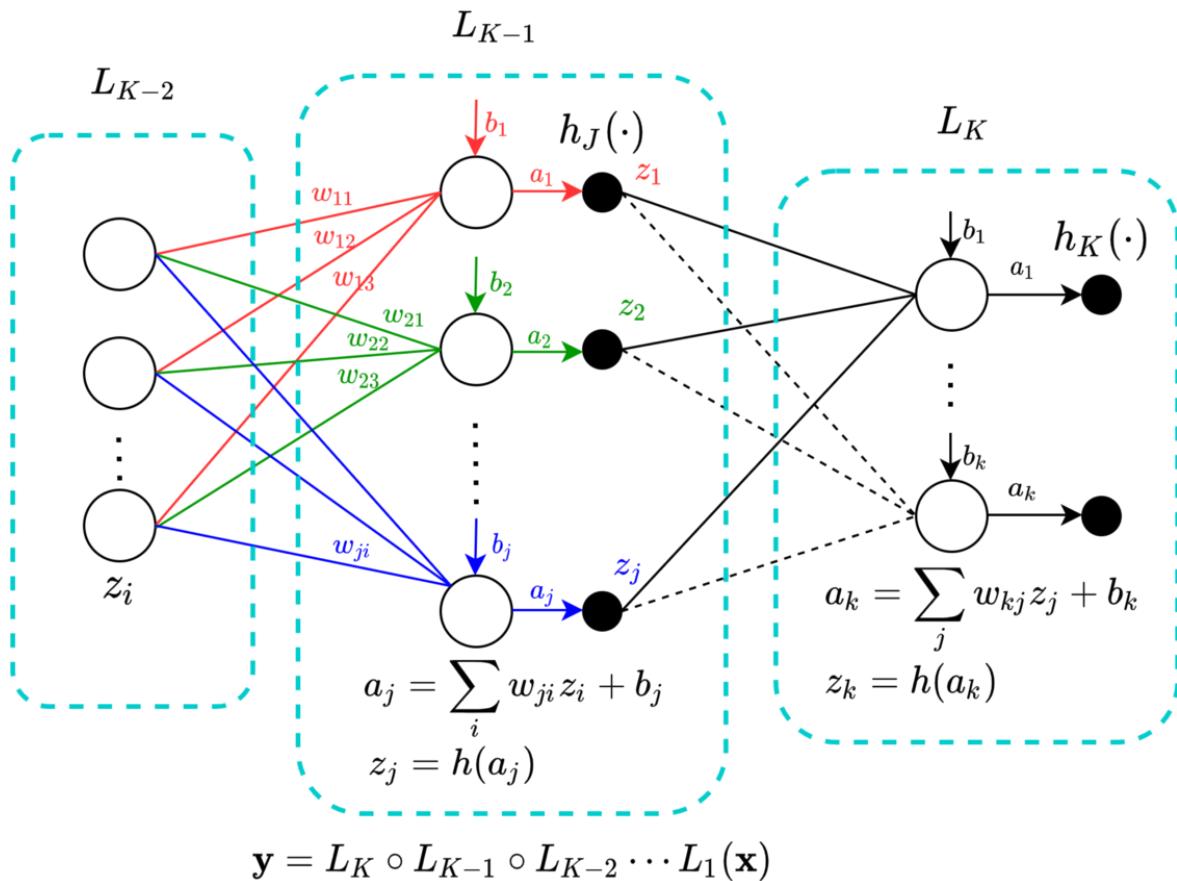
### 3.4.1 Neat Algoritme

Neat oftewel Neuro-Evolution of Augmenting Topologies is een populatie-gebaseerd evolutionair algoritme ontwikkeld door Kenneth Stanley in 2002. Het algoritme is gebaseerd op twee belangrijke principes:

- Concurrerende conventies vermijden via historische markeringen: Het kan gebeuren dat twee individuen hetzelfde gedrag laten zien terwijl ze totaal verschillende genotypes hebben. Dit leidt tot slechtere volgende generaties. Neat lost dit op door historische markeringen van nieuwe structurele elementen te behouden.
- Complexificatie: Het netwerk begint met een zo eenvoudig mogelijk netwerk en het algoritme voegt alleen nieuwe structurele elementen toe zoals neuronen verbindingen.

### 3.4.2 Backpropagation

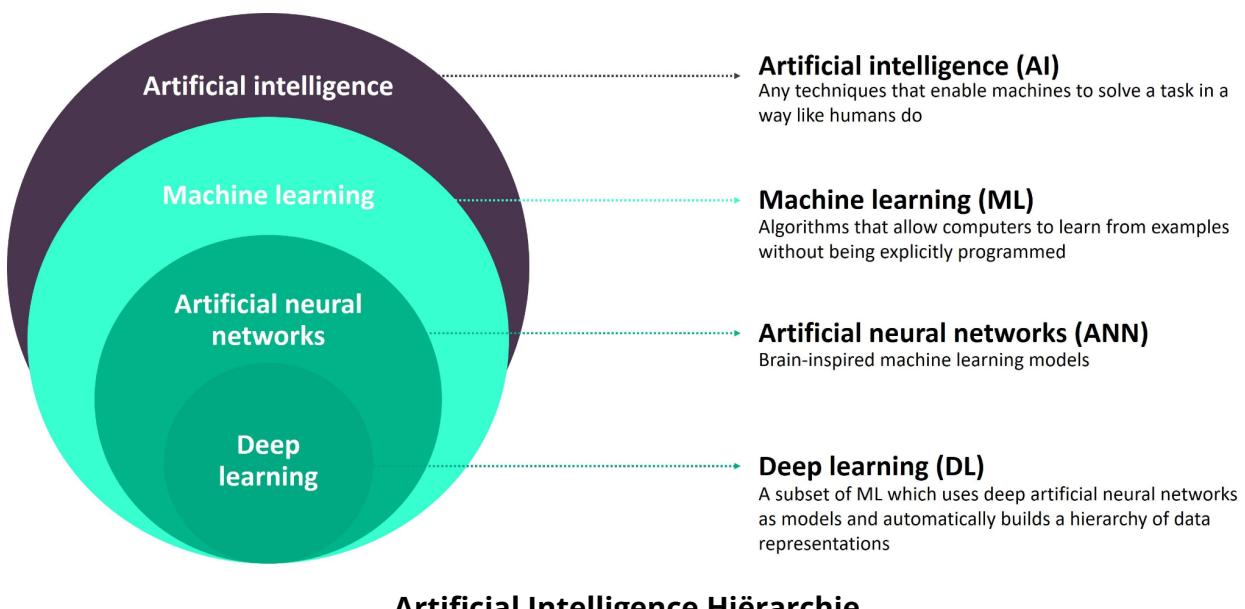
Backpropagation is een supervised algoritme dat wordt gebruikt voor het trainen van neurale netwerken. Het gebruikt gradiënt afdaling of delta regel om te zoeken naar de minimale fout functies in de gewichtsruimte en selecteert gewichten die de foutoplossing minimaliseren. Het is een methode om de gewichten van het neurale netwerk nauwkeurig af te stemmen op basis van het foutenpercentage, dat in de vorige eeuw verkregen is. Foutpercentages worden verminderd door de juiste afstemming van gewichten, wat ook de generalisatie van het model vergroot. Onze A.I. wordt met dit algoritme getraind. Aan de hand van informatie kan de A.I. zichzelf verbeteren. We laten de auto een bepaalde route rijden en verzamelen vervolgens data, zoals de snelheid, de rotatie en de afstand tot zijn omgeving. De A.I. probeert dan weights en biases zo aan te passen, dat de output van de A.I. heel dicht in de buurt komt bij wat er verwacht wordt.



### 3.4.3 A.I. Hiërarchie

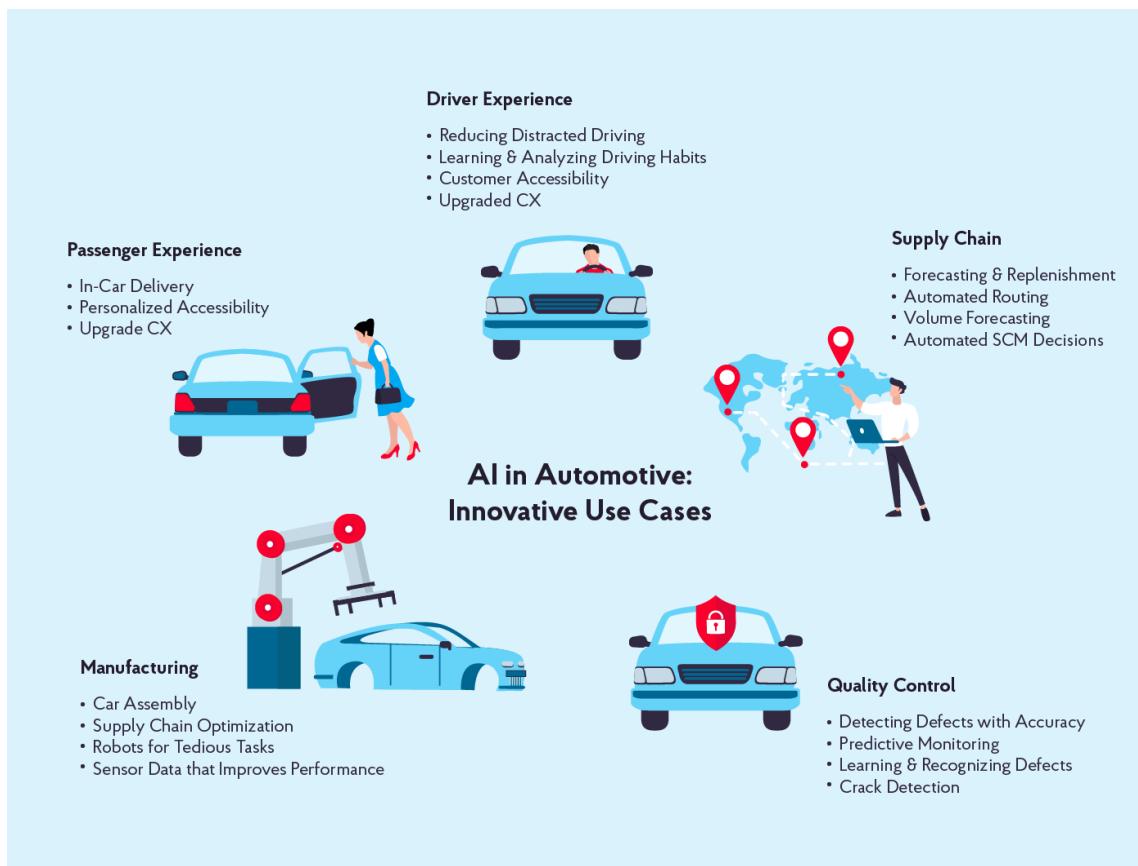
Hieronder beschrijven wij de hiërarchie van artificial intelligence:

- **Machine learning:** Machine learning is het maken van algoritmen, die uit voorbeelden en ervaringen kunnen leren. Machine learning is gebaseerd op het idee dat sommige patronen in de data geïdentificeerd en gebruikt kunnen worden voor toekomstige voorspellingen.
- **Artificial neural networks:** ANN is een rekenkundig leersysteem dat een netwerk van functies gebruikt om de input data te begrijpen en te vertalen naar de gewenste output data, dit is meestal in een andere vorm. Het concept van het kunstmatige neurale netwerk is gebaseerd op de manier waarop neuronen van het brein samenwerken om de data die de zintuigen verzenden te begrijpen.
- **Deep learning:** Deep learning is een machine learning techniek die is gebaseerd op de manier waarop mensen van nature leren door het juiste voorbeeld te geven. Deep learning is een belangrijk onderdeel van zelfrijdende auto's. Met behulp van Deep Learning kunnen deze auto's een stopbord herkennen of een voetganger van een lantaarnpaal onderscheiden. De diepte van het model wordt met het aantal lagen dat het model heeft weergegeven.



### 3.4.4 A.I. In de auto industrie

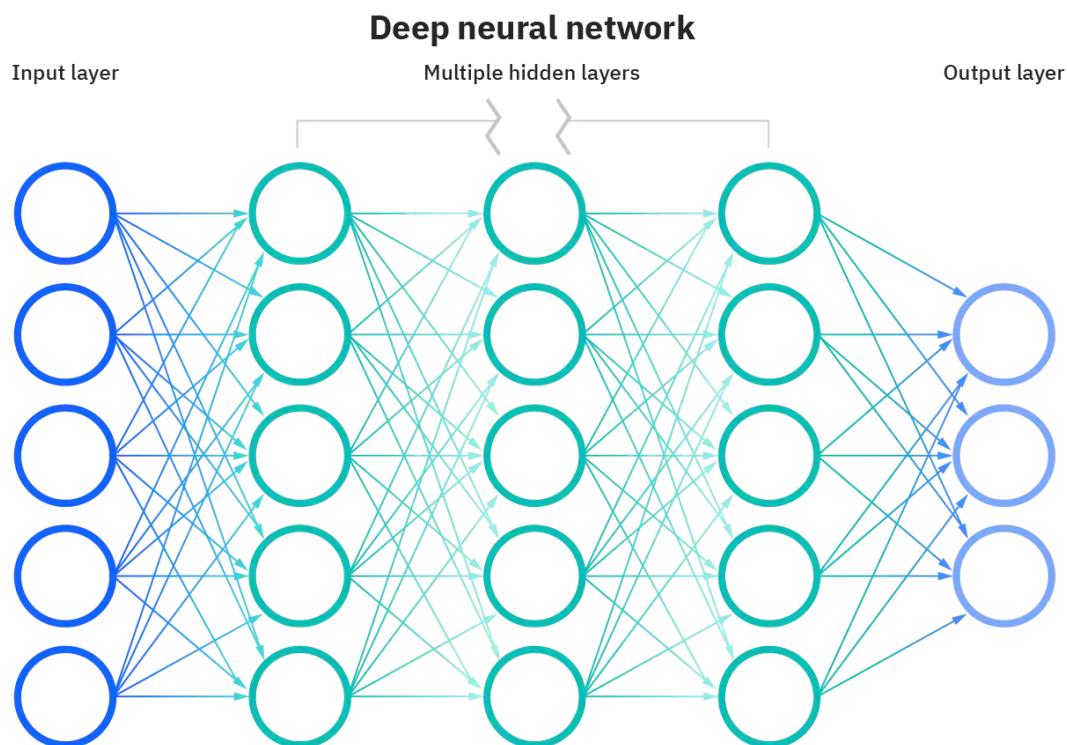
Het bedrijf Tesla is toonaangevend op het gebied van zelfrijdende auto's. In het begin richtte het bedrijf zich op het ontwikkelen van elektrische auto's voor een groenere en emissievrije planeet. Om dit langverwachte doel te bereiken, introduceerde Tesla de Autopilot. Over minder dan 10 jaar zal Autopilot zo verbeterd zijn dat je maar de bestemming aan hoeft te geven en de auto zelfstandig naar de bestemming kan rijden, zolang iemand in de auto aanwezig is om toezicht te houden over de auto. Deze behoorlijke vooruitgang is te danken aan het grote netwerk van zelfrijdende auto's die Tesla in haar bezit heeft. Elke keer dat een bestuurder moet ingrijpen om de Autopilot te corrigeren, wordt de verzamelde data naar Tesla teruggestuurd om de A.I. te verbeteren en een dergelijke situatie in de toekomst te voorkomen. Door de grootte van het netwerk heeft Tesla de mogelijkheid om de zelfrijdende A.I. op veel gevareerde en realistische situaties te trainen. Hoewel dit volgens Tesla de beste trainingsmanier is, wordt dit niet door iedereen in de industrie gebruikt. A.I. wordt vaker gebruikt in de auto-industrie. Robots verantwoordelijk voor de productie van auto's worden steeds zelfstandiger.



## 3.5 Neuraal Netwerk

### 3.5.1 Layers

Een A.I. bestaat uit een neuraal netwerk dat telkens wordt aangepast om zo goed mogelijk te presenteren. Een voorbeeld van een neuraal netwerk is zichtbaar in de afbeelding hieronder. Neurale netwerken bestaan uit neuronen, verbindingen tussen neuronen en biases, waarvan de verbindingen tussen de neuronen en biases een waarde hebben die bij het neurale netwerk horen. De neuronen zijn de cirkels in de afbeelding en de verbindingen zijn de lijnen die deze cirkels verbinden. Een biases is een waarde die meegegeven aan een neuron. Door de waarden van de biases en connecties aan te passen leert de AI. Alle waarden samen bevatten dus de informatie die nodig is om de A.I. iets te laten doen. Je kunt een A.I. alles laten doen wat je maar wilt. Dus het klinkt misschien vreemd dat een aantal waarden al deze informatie kunnen bevatten, maar de hoeveelheid aan waardes in je neuraal netwerk neemt heel snel toe: Een neuraal netwerk heeft al gauw duizenden waarden.



---

## Sigmoïdefunctie

Een neuraal netwerk is opgebouwd uit meerdere soorten lagen: de input layer, de hidden layers en de output layer. De inputs van het neuraal netwerk worden gegeven in de input layer en in de output layer geeft het neuraal netwerk zijn outputs. Daartussen zitten de hidden layers die al het denken doen. Elke layer wordt verbonden met de layer ervoor door connecties, elke neuron heeft een formule om gerekend te worden.

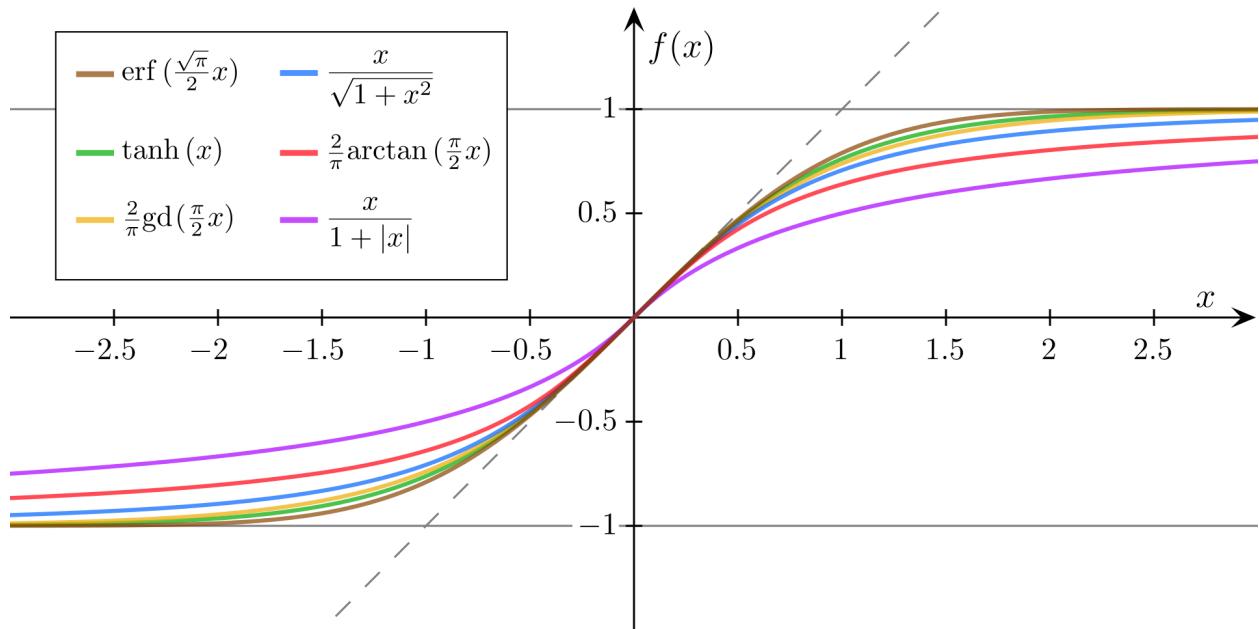
Elke neuron noteerden wij als  $a_c^{(d)}$ , waarin d de layer van de neuron is en c de rij van de neuron. Elke neuron heeft ook een bias die op dezelfde manier genoteerd wordt maar met een a in plaats van een b:  $b_c^{(d)}$ . De connecties worden genoteerd als  $w_{e,f}$  waarin de rij van de neuron waar de connectie naartoe gaat is en of de neuron waar de connectie vandaan komt. De waarde van een neuron valt te berekenen als volgt:

$$a_k^{(n+1)} = \sigma(w_{k,0}a_0^{(n)} + w_{k,1}a_1^{(n)} + \dots + w_{k,p}a_p^{(n)} + b_k^{(n+1)})$$

P is hierin de lengte van layer n en σ is de sigmoid functie die gebruikt wordt om de hele getallenlijn om te zetten naar een getal tussen 0 en 1. De sigmoid functie gaat als volgt:

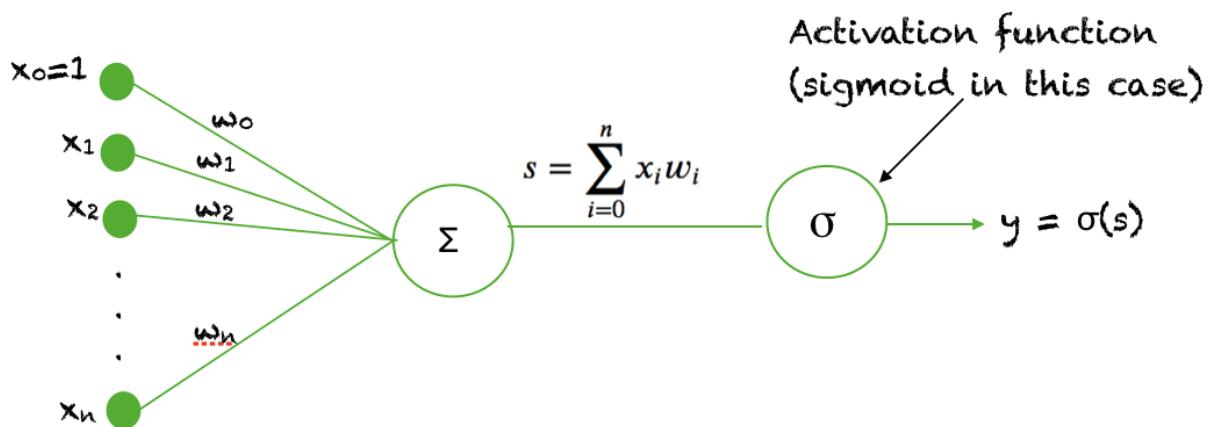
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Je telt dus alle waardes van de neurons van de vorige layer keer de bijbehorende waardes van de verbindingen bij elkaar op. Daarbij tel je ook nog de bias op van de neuronen die je wilt berekenen en tot slot stop je die uitkomst in de sigmoid functie, die er een getal van 0 tot 1 van maakt.



**Enkele sigmoïde functies vergeleken**

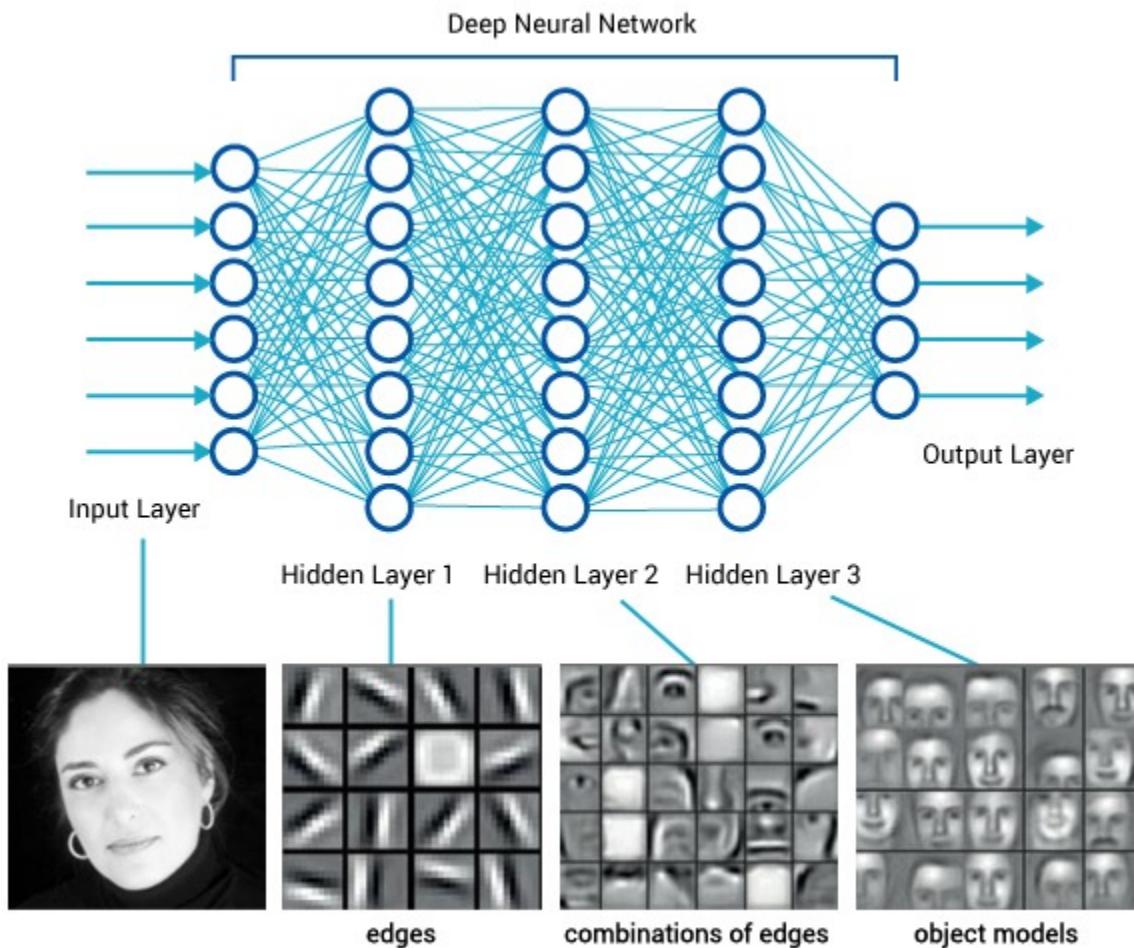
De sigmoïde functie wordt gebruikt als activeringsfunctie in neurale netwerken. De onderstaande afbeelding toont de rol van een activeringsfunctie in één laag van een neuraal netwerk. Een gewogen som van input wordt toegepast in een activeringsfunctie en de output ervan wordt als input voor de volgende laag gebruikt.



**Sigmoïde in een neuraal netwerk**

## Layer abstracties

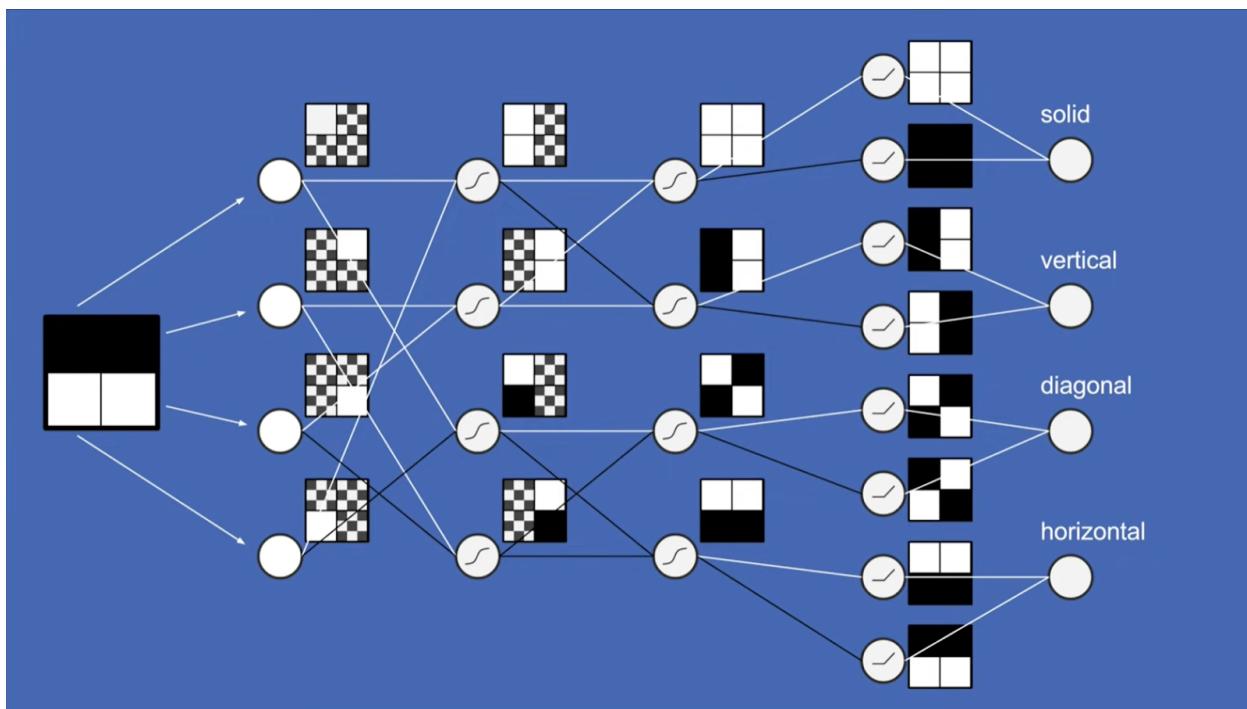
Het is voor een mens lastig te begrijpen hoe een neuraal netwerk exact een taak uitvoert, omdat er zoveel gebeurt in een neuraal netwerk en neurale netwerken ook niet op dezelfde manier denken als mensen. Wel kunnen we een voorbeeld bedenken van hoe het zou kunnen gaan. In de afbeelding hieronder wordt dit weergegeven. In de afbeelding wordt zichtbaar gemaakt hoe er per layer naar steeds complexere patronen wordt gezocht. Elk groter patroon is gebaseerd op de patronen in de layer ervoor. In de afbeelding wordt zo late zien dan je vanaf pixels kan zoeken naar randen en uit groepen randen kunt zoeken naar bijvoorbeeld ogen, monden en neuzen. En vervolgens vormt de AI hieruit gezichten.



Lagen en hun abstractie in een diep neuraal netwerk

Ook kunnen we een wat exakter voorbeeld nemen, zoals in de afbeelding hieronder. Dit is een AI die uit blokken van 2x2 pixels kan herkennen of het een blok, horizontale lijnen, verticale lijnen of diagonale lijnen vormt. In het voorbeeld betekent witte lijnen positieve connecties, zwarte lijnen betekenen negatieve en maken we geen gebruik van de sigmoid functie. In plaats daarvan delen we de uitkomst door twee, want dat maakt de berekening makkelijker.

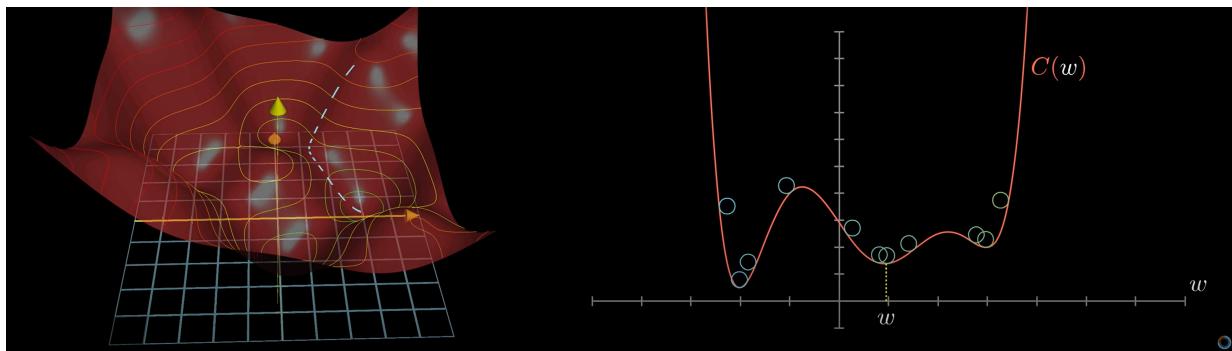
De eerste laag is de input layer. Deze kijkt naar de afbeelding en gaat per pixel na welke kleur hij is. Bij wit wordt de input op 1 gezet en bij zwart op -1. Dus de bovenste van deze lagen geeft -1 want dat vak in de voorbeeldafbeelding is zwart en de onderste van de lagen geeft 1. Dan gaan we door naar de volgende layer. Om bijvoorbeeld de waarde van de bovenste van de tweede layer te weten, moet je de waarde van de input neurons keer de waarde van de connecties doen die naar deze neuron gaan en dit delen door twee. In dit geval is dat de bovenste uit de input layer plus de onderste gedeeld door twee.  $(1-1)/2=0$ , dus de waarde van deze neuron is 0. Als je dit verder berekent voor alle neurons zal de waarde van de horizontale neuron in de output layer 1 zijn.



**Pixel detectie**

### 3.5.2 Lokale optima

Elke opdracht van een neuraal netwerk kan gezien worden als een grote functie met een bepaalde hoeveelheid inputs en outputs. Aan de outputs kan een score worden gegeven van hoe goed de outputs zijn, het doel van de A.I. is deze score te optimaliseren. Dit wordt gedaan door de waarden van de connecties en biases aan te passen. In de linker afbeelding hieronder staat een voorbeeld van hoe zo'n functie eruit zou kunnen zien met maar twee aanpasbare waarden, in de werkelijkheid zijn dit er duizenden. Als het aan de A.I. is om de score van de outputs zo laag mogelijk te maken, moet de A.I. de waardes vinden van het laagste dal van de twee dalen zichtbaar in de afbeelding.



Een A.I. doet dit door op een random plek op de functie te beginnen en vervolgens proberen de richting naar beneden te vinden in de functie. De functie gaat vervolgens zijn waardes zo aanpassen dat het punt een klein stukje in die richting beweegt. Je kan dit voorstellen als een bal die losgelaten wordt ergens in de functie. De beweging is hetzelfde: de bal rolt in de richting waar het het steilst naar beneden gaat. Na een aantal keer een stap naar beneden genomen te hebben komt de bal uiteindelijk in een dal terecht. Dit hoeft echter niet het minimum van de functie te zijn. De rechter afbeelding is een voorbeeld van hoe dat eruit zou kunnen zien. Het kan namelijk zijn dat de bal bij het lokale minimum  $\omega$  beland en niet in het echte minimum dat links zit. Om dit te voorkomen begint de A.I. van een hoop verschillende random startlocaties. Dit maakt de kans groter, maar nooit zeker dat het beste minimum gevonden wordt.

---

## H4 Deelvragen

Om dit onderzoek te uitvoeren en antwoord te vinden op de hoofdvraag, moeten de volgende vragen worden beantwoord. Het proces van ontwikkeling van de auto wordt in vier delen opgedeeld: [Kunstmatige Intelligentie](#), [Simulatie](#), [Auto](#) en [Werkelijkheid](#).

### 4.1 Kunstmatige Intelligentie

Om onze auto zelfrijdend te maken zullen wij gebruik maken van een evolutionair neuraal netwerk, wat een neuraal netwerk is dat gebruik maakt van neuronontwikkeling.

#### 4.1.1 Wat houdt Artificial Intelligence in?

A.I. is een tak van de informatica met als hoofddoel het maken van intelligente machines. Deze machines moeten in staat zijn taken uit te voeren, waarvoor menselijk verstand nodig is. A.I. is een simulatie van menselijke intelligentie door de computer. A.I. is dus eigenlijk een machine die is geprogrammeerd, die het vermogen heeft om menselijk gedrag te kopiëren en die net zo slim kan denken als een mens. Deze definitie van A.I. kan op alle machines, die op dezelfde manier functioneren als menselijk verstand en taken kunnen uitvoeren, zoals echt oplossen van problemen, leren en automatiseren, worden toegepast.

#### 4.1.2 Wat zijn de voordelen van zelfrijdende auto's?

Zelfrijdende auto's zullen veel voordelen met zich meebrengen in de toekomst. Een zelfrijdende auto is bijvoorbeeld veiliger dan door personen bestuurde auto's, omdat computers een schnellere reactietijd, van ongeveer 10 milliseconde, hebben, terwijl die van een gemiddelde volwassen mens ongeveer 250 milliseconde is. Daarnaast kunnen de verschillende sensoren in de zelfrijdende auto's verkeerssituaties beter inschatten en snel daarop reageren. Een ander voordeel is dat er minder files zullen ontstaan, aangezien computers taken efficiënter en veiliger kunnen uitvoeren dan mensen. Zelfrijdende auto's zijn nog maar het begin. Uiteindelijk maakt A.I. een computer zo intelligent dat de computer beslissingen kan nemen, die mensen nooit zouden kunnen nemen. Voorbeelden van toepassingen van A.I. in onze samenleving zijn ingewikkelde kankeronderzoeken, beeldherkenning en gepersonaliseerde marketing.

#### [Autonomoos rijden in stroomversnelling door sensortechniektechniek](#)

---

### **4.1.3 Welke methoden van dataverzameling zijn het meest geschikt voor zelfrijdende auto's?**

Het is voor ons project belangrijk dat de zelfrijdende auto overal om zich heen kan kijken, zodat hij goed kan reageren op zijn omgeving. Daarnaast moet hij ook objecten kunnen herkennen, om te weten hoe hij op verschillende objecten moet reageren. De auto moet bijvoorbeeld wel stoppen voor een persoon die voor de auto loopt, maar niet voor een vogel die voor de sensor langs vliegt of voor andere auto's die in de linker of rechter rijstrook zijn. De auto moet dit zonder input van de bestuurder doen en zelfstandig kunnen rijden. De meest belovende dataverzameling methoden voor zelfrijdende auto's zijn camera en lidar of radar. Voor de dataverzameling hebben wij dus de keuze tussen camera, Lidar scanner, ultrasonic geluid sensor of een combinatie daarvan.

Voor het herkennen van objecten en obstakels en beslissingen nemen is een camera een goede keuze. Hierdoor kan de auto alles herkennen en overal om zich heen kijken. De camera ziet dus de omgeving zoals een persoon dat zou zien, en de A.I. denkt ook zoals een persoon zou denken met behulp van dat beeld. Dit geeft ons meer mogelijkheden voor het ontwikkelen van een A.I. die juist functioneert. De ontwikkelingen van op camera gebaseerde zelfrijdende auto's gaan snel, doordat [\*\*Semiconductor device fabrication\*\*](#) techniek steeds verbeterd, waardoor bedrijven kleinere, efficiëntere en snellere chips produceren die minder kosten. Lidar tast de omgeving af met licht (laser of infrarood). Radar doet hetzelfde, maar dan met radiogolven. Bij lidar en radar wordt er van de techniek [\*\*Remote sensing\*\*](#) gebruikgemaakt. Het voordeel van lidar ten opzichte van radar is de hoge resolutie, wat nodig is om nauwkeurig stilstaande en bewegende objecten te kunnen detecteren. Een nadeel is dat weersomstandigheden, zoals mist en regen, een grote negatieve invloed op de nauwkeurigheid van Lidar hebben. Lidar is geschikt voor het waarnemen van bewegende objecten in de directe nabijheid van een voertuig. Radar kan verder kijken, maar met het toenemen van de afstand neemt de nauwkeurigheid af. Daarom is radar meer geschikt voor het op afstand waarnemen van bewegende objecten.

#### [\*\*Sensors - LiDAR vs. RADAR\*\*](#)

Tesla gebruikt Tesla Vision in plaats van een radar in haar auto's om zelfstandig te kunnen rijden. In onze auto gaan wij daarom een combinatie van een camera en een Lidar scanner gebruiken, zodat wij ook dichtbij de werkelijkheid kunnen komen.

---

#### **4.1.4 Welk library gebruiken wij voor het opbouwen van ons neurale netwerk?**

We gaan ons A.I. met behulp van tensorflow opbouwen. TensorFlow maakt het heel makkelijk om een neuraal netwerk op te bouwen dat heel efficiënt is. Het biedt genoeg verschillende functies aan die ons helpen in het proces van het ontwerpen van onze A.I. Er zijn functies waarbij we het model kunnen traceren gedurende zijn training. Als we onze eigen A.I. hadden gemaakt, zou het heel complex, lang, tijdrovend en inefficiënt geweest zijn.

#### **4.1.5 Hoe worden de beste auto's geselecteerd?**

Om onze A.I. te trainen gebruiken wij `model.fit_generator()` functie van de tensorflow library. Het neuraal netwerk wordt voor een bepaald aantal epochs getraind. Bij elke epoch vindt er backpropagatie plaats. Dat wil zeggen dat de weights en biases van het neuraal netwerk zo worden aangepast, dat de loss waarde, die geeft aan hoe ver de voorspelde waarde is van de gewenste waarde, nul benadert.

```
model.fit_generator(image_data_generator(x, y, batch_size),
                    steps_per_epoch=int(len(x) // batch_size),
                    epochs=epochs,
                    verbose=1,
                    validation_data=image_data_generator(x, y, batch_size),
                    validation_steps=int(len(x) // batch_size))
```

Hier trainen wij het neurale netwerk voor 10 epochs. Steps\_per\_epoch is het één keer doorgaan van backpropagation op één batch, die in dit geval 500 is.

## 4.1.6 Hoe slaan wij het neurale netwerk op om de auto te laten rijden?

Als het aantal epochs doorgelopen is, wordt het beste neurale netwerk met laagste loss waarde opgeslagen.

```
model.save(path_to_save)
```

## 4.1.7 Hoe bepalen wij de structuur van het neurale netwerk?

Het neurale netwerk moet minstens een input-layer en een output-layer bevatten. Het aantal neuronen in de input layer is gelijk aan het aantal hoeken waaronder wij de afstand van de auto meten tot zijn omgeving. Hoe meer hoeken, hoe nauwkeuriger de auto rijdt. Dat wil zeggen dat de auto minder vaak tegen zijn omgeving botst. Het aantal neuronen in de output layer is gelijk aan twee, namelijk wat de snelheid is en wat de steering angle van de auto moet zijn. Onze hidden layers hebben wij willekeurig gekozen. Als het een complex project was, hadden wij veel complexere hidden layers moeten kiezen.

```
class NeuralNetwork:
    def __init__(self, structure):
        self.structure = structure
        self.model = tf.keras.Sequential()

        self.model.add(tf.keras.layers.Input(structure[0]))
        for i in range(len(structure)):
            if i < len(structure) - 1:
                self.model.add(tf.keras.layers.Dense(structure[i], activation='tanh'))
            else:
                self.model.compile(loss=tf.keras.losses.MeanAbsoluteError(),
metrics=['accuracy'])
```

---

#### **4.1.8 Hoe kunnen wij ervoor zorgen dat de zelfrijdende auto op een veilige manier rijdt?**

Als wij het hebben over de veiligheid van zelfrijdende auto's zijn er meerdere factoren van toepassing. Namelijk de vaardigheden van de programmeur, de nauwkeurigheid van de sensoren en de snelheid van de computers die gebruikt worden in de auto's. Een goed geprogrammeerde A.I. die goed de omgeving bekijkt en opslaat, is absoluut geschikt om veilig te rijden.

Om de A.I. goed te trainen, herhalen wij het trainen een aantal keer, totdat wij tevreden zijn met het resultaat. Daarnaast hebben wij geprobeerd de meest geschikte onderdelen te kiezen en te gebruiken voor de robot, zodat wij juiste en nauwkeurige metingen hebben voor het trainen van de AI. De Lidar-scanner berekent de tijd tussen het moment dat de lichtstralen een object of oppervlakte raken en het moment dat de lichtstralen weer worden teruggekaatst naar de laserscanner. De afstand wordt daarna berekend met behulp van de lichtsnelheid. Op deze manier hebben wij een 3D realtime beeld van de omgeving van de robot en kunnen wij de A.I. zo programmeren dat hij de obstakels 'ziet' en de juiste beslissingen kan nemen.

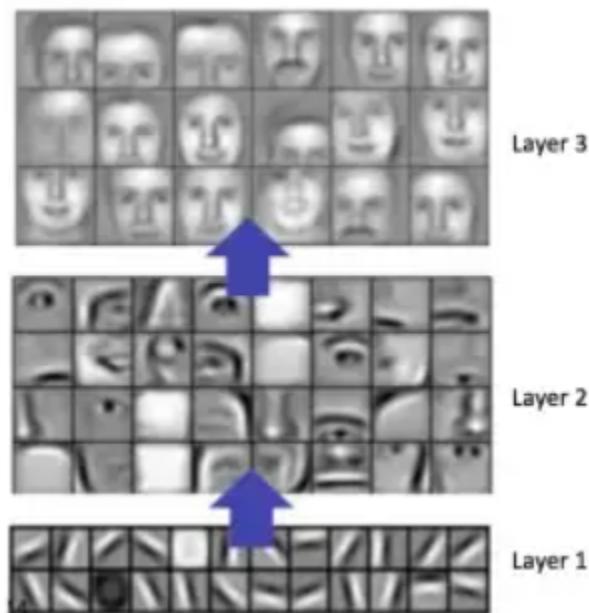
#### **4.1.9 Kan de A.I. zijn omgeving begrijpen en herkennen?**

Ook is het mogelijk om van meerdere sensoren gebruik te maken. Een camera is een geschikt middel hiervoor. De A.I. kan met behulp van een camera niet alleen zijn afstand tot zijn omgeving bepalen, maar het kan ook zijn omgeving zien, begrijpen en daarop reageren. De A.I. kan bijvoorbeeld getraind worden om verschillende verkeersborden te herkennen en daarop volgens de wet te reageren. Echter het herkennen van verkeersborden was niet een doel voor ons. We hebben ervoor gekozen dat de A.I. een rijbaan moet kunnen volgen. Er zijn namelijk in werkelijkheid geen muren die herkend kunnen worden met een lidar sensor rondom de auto en met een lidar sensor kan ook geen rijbaan herkend worden.

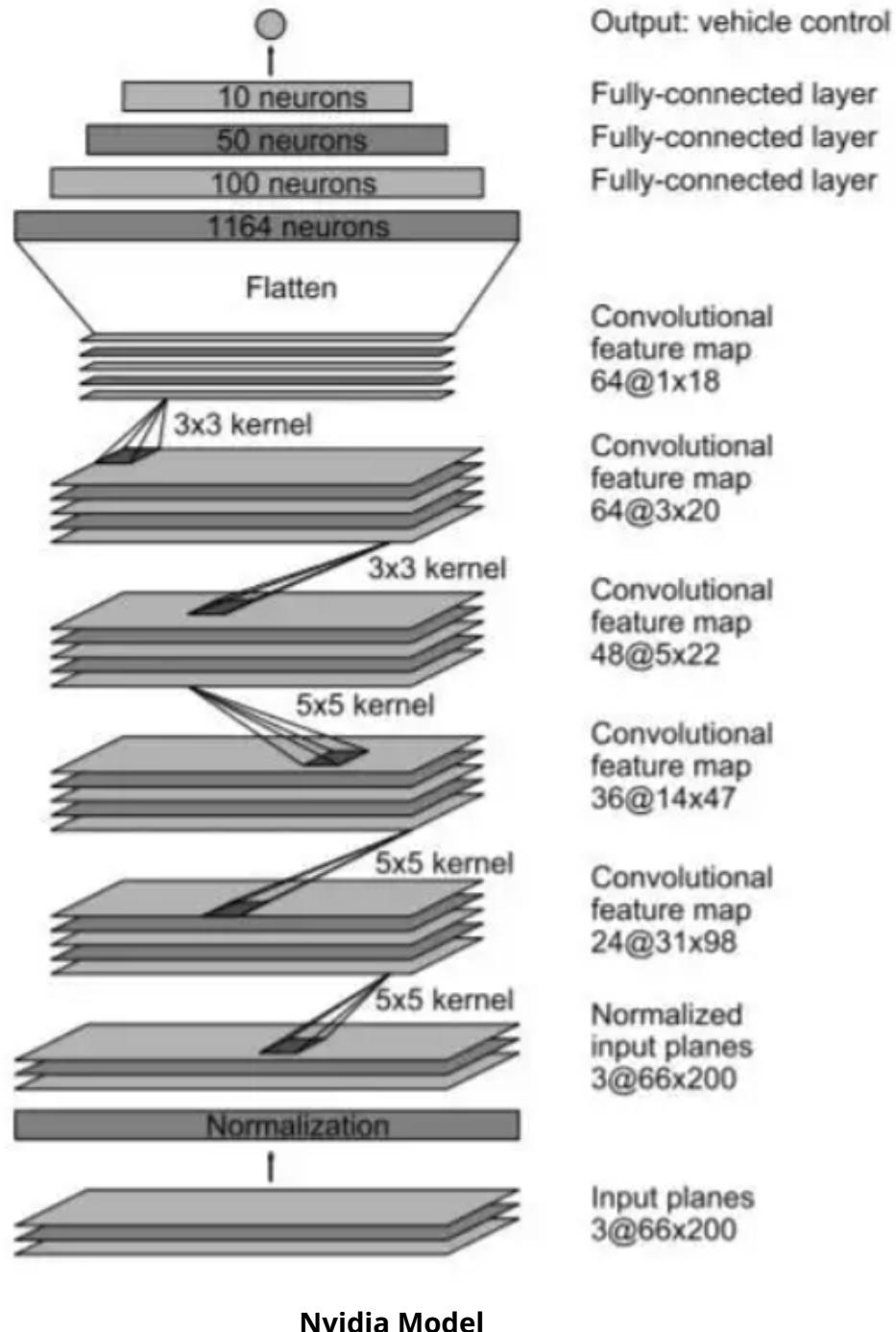
## Nvidia model

Op het hoogste niveau zijn de inputs voor het Nvidia-model videobeelden van DashCams aan boord van de auto, en outputs zijn de stuurhoek en hoeveelheid gas van de auto. Het model gebruikt de videobeelden, haalt er informatie uit en probeert de stuurhoeken en hoeveelheid gas van de auto te voorspellen. Dit staat bekend als een gecontroleerd machine learning-programma, waarbij videobeelden (kenmerken genoemd) en stuurhoeken (labels genoemd) worden gebruikt in de training. Omdat de stuurhoeken en hoeveelheid gas numerieke waarden zijn, is dit een regressieprobleem in plaats van een classificatieprobleem, waarbij het model moet voorspellen of het een hond of een kat is, of welke soort bloem in de afbeelding voorkomt.

De kern van het NVidia-model is een Convolutional Neural Network. CNN's worden veel gebruikt in deep learning-modellen voor beeldherkenning. De intuïtie is dat CNN vooral goed is in het extraheren van visuele kenmerken uit afbeeldingen uit de verschillende lagen (ook wel filters genoemd). Voor een CNN-model voor gezichtsherkenning zouden de eerdere lagen bijvoorbeeld basiskenmerken extraheren, zoals lijnen en randen, zouden middelste lagen meer geavanceerde kenmerken extraheren, zoals ogen, neuzen, oren, lippen, enz. of het hele gezicht, zoals hieronder afgebeeld.



De CNN-lagen die in het Nvidia-model worden gebruikt, lijken sterk op het bovenstaande, omdat het lijnen en randen in hun vroege lagen extraheert en complexe vormen in hun latere lagen. De volledig verbonden lagen functie kan worden gezien als een functie voor besturing. Het bovenstaande diagram is overgenomen van Nvidia's onderzoek.



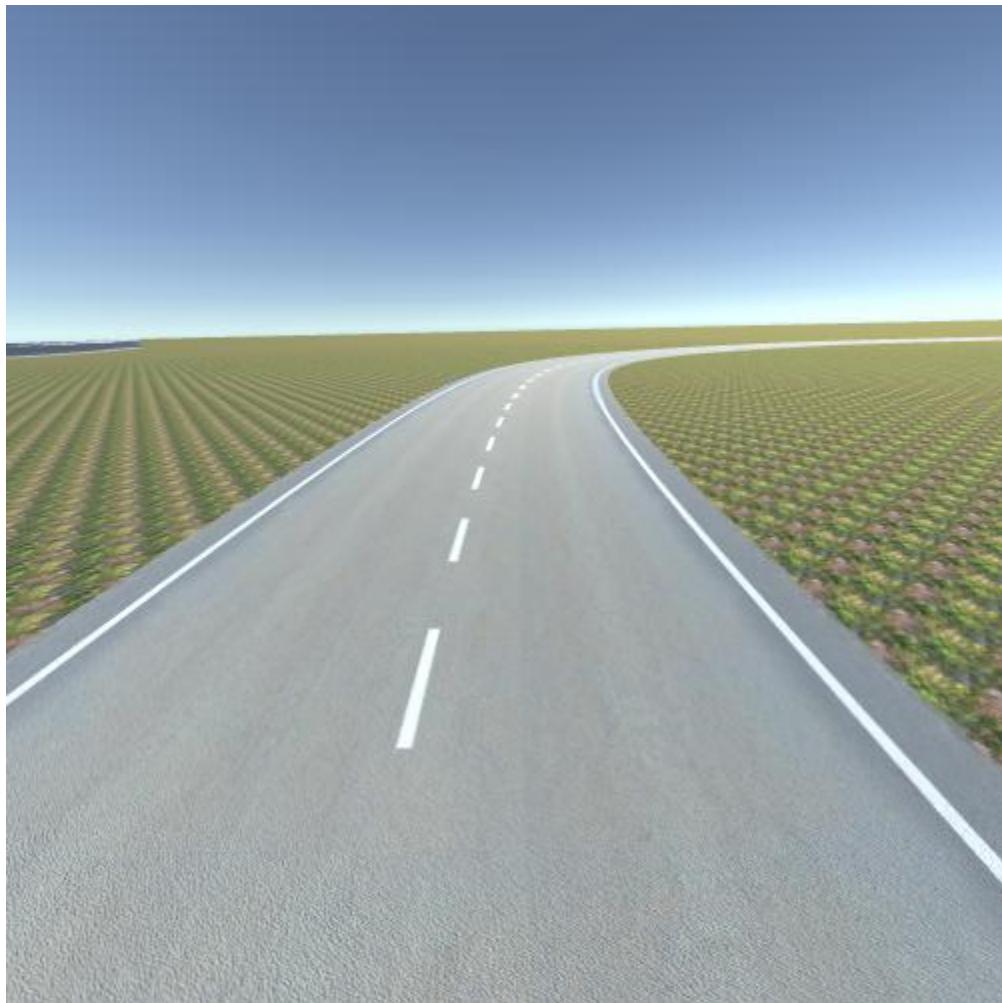
---

Het bevat in totaal ongeveer 30 lagen. De invoer afbeelding voor het model (onderaan het diagram) is een afbeelding van  $66 \times 200$  pixels, maar later gebruiken wij een  $250 * 250$  pixels input omdat het A.I. hierdoor beter leerde. Het beeld wordt eerst genormaliseerd, vervolgens door 5 groepen convolutionele lagen geleid, uiteindelijk door 4 volledig verbonden neurale lagen geleid en tot twee outputs geleid, namelijk de door het model voorspelde stuurhoek en hoeveelheid gas van de auto. Deze door het model voorspelde hoek en gas wordt vervolgens vergeleken met de gewenste stuurhoek en hoeveelheid gas gegeven in het videobeeld, de fout wordt via backpropagation teruggekoppeld naar het CNN-trainingsproces. Zoals te zien is in het bovenstaande diagram, wordt dit proces in eenlus herhaald totdat de fouten (ook wel verlies of Mean Squared Error genoemd) laag genoeg zijn, wat betekent dat het model heeft geleerd redelijk goed te sturen. Ons simulatie is een belangrijk onderdeel hiervan omdat het de training data oplevert die wij gaan gebruiken voor het trainen van het A.I. door backpropagation.

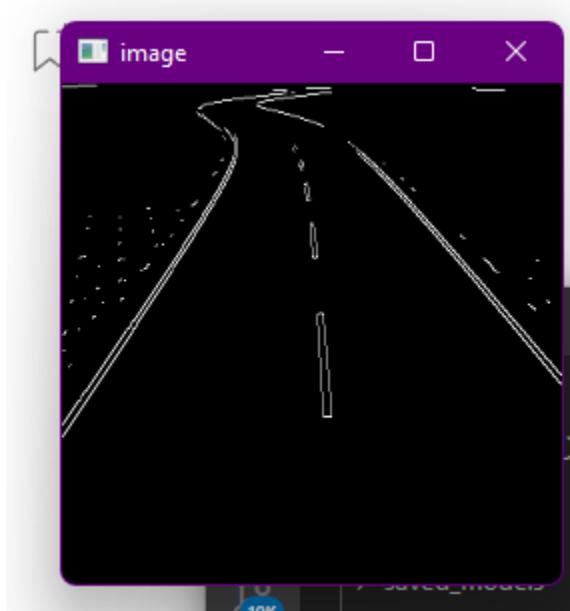
---

## Image preprocessing

Hieronder zijn twee afbeeldingen uit de simulatie te zien. Een normale afbeelding die door de camera van het auto wordt opgenomen en een zwart en wit afbeelding die de afbeelding is wat het A.I. daadwerkelijk ziet.



**Unprocessed image**



**Processed image**

We hebben de A.I. eerst met de normale afbeeldingen getraind maar na enige tijd was de A.I. niet in staat om zelfstandig te rijden. De oorzaak hiervan zijn de onbelangrijke data, zoals de vegetatie rond de auto, die invloed heeft op de besluitvorming van de A.I. De enige informatie die de A.I. nodig heeft om de auto te besturen, is de lijnbaan. Om in de originele afbeelding de twee lijnbanen te herkennen gebruiken wij OpenCV library in python. OpenCV is een bibliotheek met programmeerfuncties die voornamelijk gericht zijn op realtime computervisie. Om het verwerkte afbeelding te krijgen, gebruiken wij maar de helft van de hoogte van de afbeelding zodat het horizon niet als een rijbaan wordt afgebeeld. Dan gebruiken wij enigszins blur op de afbeelding zodat de randen van lijnbanen meer duidelijk worden. Dan wordt de afbeelding zwart en wit gemaakt die nodig is voor de lane detection functie van het OpenCV library. Deze zwart en witte afbeelding wordt doorgegeven aan het cv2.canny filter, die de randen van het afbeelding, waaronder lijnbaan, duidelijk maakt en andere onbelangrijke data zoals de vegetatie als zwart weergegeven. Echter is het nog niet klaar. Ons neurale netwerk krijgt een  $250 * 250 * 3$  pixel afbeelding binnen, wat niet overeenkomt met een  $250 * 250$  pixel zwart en wit afbeelding. De drie bij  $250 * 250 * 3$  is bedoeld voor een drie channel image. Namelijk rood, groen of zwart. Bij een zwart en wit afbeelding is echter geen channel aanwezig. Het is tevens zwart óf wit. Daarom wordt de afbeelding omgezet naar de gewenste shape met behulp van numpy.reshape() functie. Vervolgens worden de pixels gedeeld met 255 om alle

---

waarden van 1 tot 255 te mappen naar 0 tot 1, die ook belangrijk is voor de AI. Het voorkomt dat er te grote getallen worden voorspeld en het trainen van het neuraal netwerk heel veel langer duurt. Ons neurale netwerk maakt gebruik van een activation functie die belangrijk is voor het oplossen van complexe problemen. We begrijpen zelf nog niet helemaal welke voordelen het oplevert, maar het artikel [What, Why and Which?? Activation Functions](#) geeft een korte uitleg over verschillende activation functies en de nut daarvan. Onze A.I. maakt gebruik van Elu activation function. De code hiervoor is hieronder te zien.

```
def img_preprocess(image):
    height, _, _ = image.shape
    image = image[int(height/3):,:,:]
    image = cv2.GaussianBlur(image, (3,3), 0)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = cv2.Canny(image, 200, 220)
    image = cv2.resize(image, (250,250))
    image = image.reshape(250, 250, 1)
    image = image / 255 # normalizing
    return image
```

### Image preprocessing

---

## Maken van het model

Het model wordt volgens het Nvidia model gemaakt met het enige verschil dat het een andere input en output vorm bevat.

```
def create_model():
    model = Sequential()
        model.add(Conv2D(24, (5, 5), strides=(2, 2),
input_shape=(250, 250, 1), activation='elu'))
        model.add(Conv2D(36, (5, 5), strides=(2, 2),
activation='elu'))
        model.add(Conv2D(48, (5, 5), strides=(2, 2),
activation='elu'))
        model.add(Conv2D(64, (3, 3), activation='elu'))
        model.add(Conv2D(64, (3, 3), activation='elu'))

    model.add(Flatten())
    model.add(Dense(100, activation='elu'))
    model.add(Dense(50, activation='elu'))
    model.add(Dense(10, activation='elu'))

##output layer:
model.add(Dense(2))

    model.compile(Adam(learning_rate=0.001), loss =
'mse', metrics = [ 'accuracy'])
return model
```

## Model maken

---

Vervolgens rijden we de auto in de simulatie en verzamelen we de data hierover. Vervolgens trainen wij het A.I. met deze data. Na ongeveer 5 minuten te trainen, kan het A.I. heel makkelijk door moeilijke routes rijden.

## 4.2 Simulatie

De eerste populatie van de neurale netwerken begint met willekeurige genes. Hierdoor presenteren deze neurale netwerken op een totaal willekeurige manier. Om dit neurale netwerk te verbeteren, maken wij gebruik van zogenaamde behavioral cloning. Het houdt in dat wij eerst zelf de auto in de simulatie rijden en de belangrijkste data zoals de stuurhoek, gas en afstand van de auto tot de muren verzamelen. Later gaan we het A.I. met deze data met behulp van tensorflow trainen. Het simuleren maakt het veel makkelijker om veel data in korte tijd te verzamelen.

---

## **4.2.1 Welke programmeertaal gebruiken wij voor de simulatie?**

Om onze simulatie zo dicht mogelijk bij de realiteit te brengen, hebben we de simulatie in Unity gemaakt. Om een bepaalde functie toe te voegen aan de auto hebben we gebruik gemaakt van C#. De auto moet namelijk van plaats kunnen veranderen en de benodigde informatie voor de A.I. zoals afstand tot voorwerpen, snelheid en de sturing hoek op kunnen slaan. Hiervoor hebben wij een CarController.cs script toegewezen aan de auto. Met dit script komt de auto nog dichter bij de realiteit omdat er gebruik wordt gemaakt van WheelColliders. De WheelCollider is een bijzondere Collider voor geaarde voertuigen. Het heeft een ingebouwde botsing detectie, wiel fysica en een op slip gebaseerd bandenrijvingsmodel.

## **4.2.2 Hoe wordt de auto bestuurd?**

De auto besturen wij met een simpel car controller script in Unity. In ons script maken wij gebruik van WheelColliders, die een onderdeel zijn van Unity Engine. Met WheelColliders komt onze simulatie nog een stap dichter bij de realiteit, aangezien wij met WheelColliders verschillende natuurkundige simulaties zoals versnelling, weerstand van de wielen tegen het asfalt, wielophanging en nog veel meer kunnen simuleren.

Als we naar voren of achteren willen rijden, moeten wij op het 'W' of 'S' knopje drukken. Dit geeft een waarde tussen -1 en 1. Om de auto te laten rijden wordt deze waarde vermenigvuldigd met de motorkracht. Om naar links of rechts te gaan drukken wij op het 'A' of 'D' knopje. Hier krijgen we ook een waarde tussen -1 en 1 die wordt vermenigvuldigd met de maximum stuur hoek. De motor kracht en maximale stuurhoek zijn vooraf bepaald. De waardes van de inputs worden vermenigvuldigd met bepaalde variabelen die vooraf door de gebruiker te bepalen zijn, zoals motorkracht, max Steering Angle en remming kracht. Door deze variabelen kunnen wij de auto op een bepaalde manier laten rijden.

```

private void FixedUpdate() {
    if(!aiRunning) {
        GetInput();
    }

    HandleMotor();
    HandleSteering();
    UpdateWheels();
}

private void GetInput()
{
    horizontalInput = Input.GetAxis(HORIZONTAL);
    verticalInput = Input.GetAxis(VERTICAL);
    isBreaking = Input.GetKey(KeyCode.Space);
}

private void HandleMotor()
{
    frontLeftWheelCollider.motorTorque = verticalInput * motorForce;
    frontRightWheelCollider.motorTorque = verticalInput * motorForce;
    currentBreakForce = isBreaking? breakForce:0f;

    if(isBreaking)
    {
        ApplyBreaking();
    }
}

private void ApplyBreaking()
{
    frontRightWheelCollider.brakeTorque = currentBreakForce;
    frontLeftWheelCollider.brakeTorque = currentBreakForce;
    rearLeftWheelCollider.brakeTorque = currentBreakForce;
    rearRightWheelCollider.brakeTorque = currentBreakForce;
}

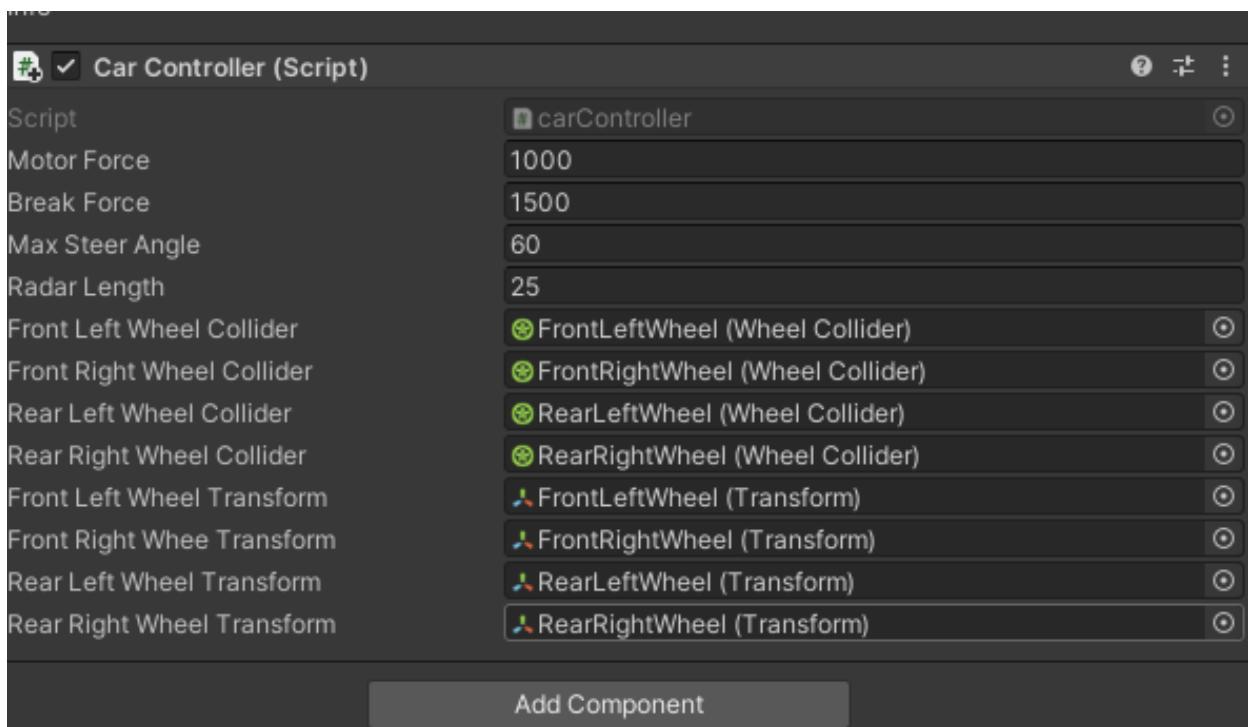
```

```
private void HandleSteering()
{
    currentSteeringAngle = maxSteeringAngle * horizontalInput;
    frontLeftWheelCollider.steerAngle = currentSteeringAngle;
    frontRightWheelCollider.steerAngle = currentSteeringAngle;
}

private void UpdateWheels()
{
    UpdateSingleWheel(frontLeftWheelCollider, frontLeftWheel);
    UpdateSingleWheel(frontRightWheelCollider, frontRightWheel);
    UpdateSingleWheel(rearLeftWheelCollider, rearLeftWheel);
    UpdateSingleWheel(rearRightWheelCollider, rearRightWheel);
}

private void UpdateSingleWheel(WheelCollider collider, Transform
transform)
{
    Vector3 position;
    Quaternion rotation;
    collider.GetWorldPose(out position, out rotation);
    transform.rotation = rotation;
    transform.position = position;
}
```

### Functions used to bestuur het auto in het simulatie



### Het instellen van de auto

#### 4.2.3 Hoe hebben wij de wereld opgebouwd?

De auto is een gebruiksklaar model dat wij van de Unity AssetStore hebben verkregen. We hebben een custom script gemaakt voor de auto om deze te laten rijden en de afstand van de auto tot de obstakels te kunnen bepalen. De afstand wordt bepaald door drie rays in verschillende hoeken uit te zenden. Als een ray tegen een object aankomt die een Mesh Collider heeft, wordt de afstand tot dit object door de Unity Engine berekend.

De wereld bevat een Plane en een aantal cubes die als obstakels dienen voor onze auto. Deze cubes hebben een Meshcollider waartegen de rays kunnen botsen.

#### 4.2.4 Hoe wordt de informatie opgeslagen?

Als de simulatie begint, worden de snelheid, steering angle en de afstanden in drie verschillende variabelen opgeslagen. Als we de simulatie afsluiten, worden de data in deze drie variabelen opgeslagen in twee tekstbestanden. De afstanden worden opgeslagen in een file genaamd inputs.txt die als input voor de A.I. dienen en de snelheid en de steering angle worden samen opgeslagen in een bestand als output voor de A.I.

```

void OnApplicationQuit()
{
    if(recordGame) {
        String outputsFilePath      =
@"C:\Users\parsa\Desktop\AI\AI_Lidar\training_data\inputs.txt";
        using(System.IO.StreamWriter file      = new
System.IO.StreamWriter(outputsFilePath))
    {
        foreach(List<float> dataEntry in outputData)
        {
            file.WriteLine(dataEntry[0].ToString() + "," +dataEntry[1]);
        }
    }
        String inputsFile      =
@"C:\Users\parsa\Desktop\AI\AI_Lidar\training_data\outputs.txt";
        using(System.IO.StreamWriter file      = new
System.IO.StreamWriter(inputsFile))
    {
        foreach(List<float> i in inputData)
        {
            for(int d=0; d<=sensorDistances.Count; d++) {
                if(d<= sensorDistances.Count) {
                    file.Write(i[d].ToString()+",");
                } else {
                    file.Write(i[d].ToString());
                }
            }
            file.WriteLine("");
        }
    }
}

```

## Data van de simulatie opslaan

## 4.2.5 Hoe lezen we de data op?

Onze A.I. is in Python geschreven. We lezen de data op in Python. Elke lijn bevat een array van een string die omgezet moet worden in een langere array, die van het float format is. Hiervoor wordt elke waarde die zich tussen een „,” bevindt opgeslagen in een array van het string formaat. Vervolgens wordt deze omgezet tot float. We zijn echter nog niet klaar. Om het A.I. sneller te trainen moeten de input waardes en de output waardes tussen 0 en 1 of tussen -1 en 1 zitten. Echter bevat onze data hee hoge en lage waardes die moeten worden omgezet naar een waarde tussen 0 en 1 voor de afstanden en tussen -1 en 1 voor de snelheid en steering angle. Hiervoor gebruiken wij deze formule:



Let the following denote:



- $S$ : input score



- $S_{min}$ : min score



- $S_{max}$ : max score



- $G_{min}$ : min gpa



- $G_{max}$ : max gpa



- $G$ : output gpa

Then your formula is:

$$G = \frac{(S - S_{min}) \cdot (G_{max} - G_{min})}{S_{max} - S_{min}} + G_{min}$$

Deze formule wordt heel vaak gebruikt in ons project en wordt bijvoorbeeld in de simulatie geïmplementeerd om de input data te kunnen lezen. De onderstaande afbeelding is een implementatie van deze formule in python:

```
def num_to_range(num, inMin, inMax, outMin, outMax):
    return outMin + (float(num - inMin) / float(inMax - inMin)) * (outMax - outMin)
```

```

def readDataInput(inputsFileName, includeSpeed=False):
    data_inputs = []
    with open(inputsFileName, "r") as textFile:
        inputs = [line.split() for line in textFile]
        for data in inputs:
            dataEntry = data[0].split(",")
            dataEntry = dataEntry[:-1]
            distances = dataEntry[:len(dataEntry)-1]
            speed = dataEntry[len(dataEntry)-1:]
            speed = [float(a) for a in speed]
            distances = [float(a) for a in distances]
            data_inputs.append(distances + speed)
    textFile.close()

    data_inputs = numpy.asarray(data_inputs)

    if(includeSpeed):
        return numpy.asarray(data_inputs)
    else:
        return numpy.asarray(data_inputs[:, :-1])

```

## Afstanden lezen

```

def readDataOutput(outputsFileName):
    data_outputs = []
    with open(outputsFileName, "r") as textFile:
        data = [line.split() for line in textFile]
        for i in data:
            a = i[0].split(",")[:-1]
            b = [float(n) for n in a]
            data_outputs.append(b)
    data_outputs = numpy.asarray(data_outputs)
    textFile.close()
    return numpy.asarray(data_outputs)

```

## Steering angle en throttle lezen

## 4.3 Auto

### 4.3.1 Welke auto gaan wij gebruiken voor ons project?

Voor ons project moesten we een auto kiezen die aan een aantal specificaties voldoet, waardoor we dachten dat hij niet te groot zou zijn, maar ook niet te klein. Als de auto te klein was, zou het lastiger worden om deze te sturen. Daarentegen, als hij te groot was, zou hij te zwaar worden, waardoor hij minder snel zou gaan. Ook moest hij makkelijk uit elkaar te halen zijn om hem te kunnen bouwen.

Na lang zoeken hebben wij de '[ROBO2106 4WD RC Smart Car Chassis met Servo](#)' gevonden die aan onze specificaties voldoet. Het nadeel van deze auto is dat hij ongemonteerd wordt verkocht, waardoor het ons wat extra tijd kostte om deze te bouwen.



**4WD RC Smart Car Chassis**

#### **4.3.2 Hoe zorgen wij ervoor dat de Lidar-sensor, Raspberry Pi en de motor genoeg stroom hebben in de auto?**

De module L298 die wij voor het besturen van de motor gebruiken, werkt op een spanning van 12V tot een maximum van 46V. Stroom over USB-poort is maximaal 5V en is niet sterk genoeg voor onze auto en is dus geen optie. De Lidar-sensor en de Raspberry Pi werken echter op een spanning van 5V, die via de Raspberry Pi 5V adapter over de USB-poort wordt opgeleverd. Deze adapter is handig wanneer wij het A.I. programma willen testen en geen gebruik willen maken van de motor.

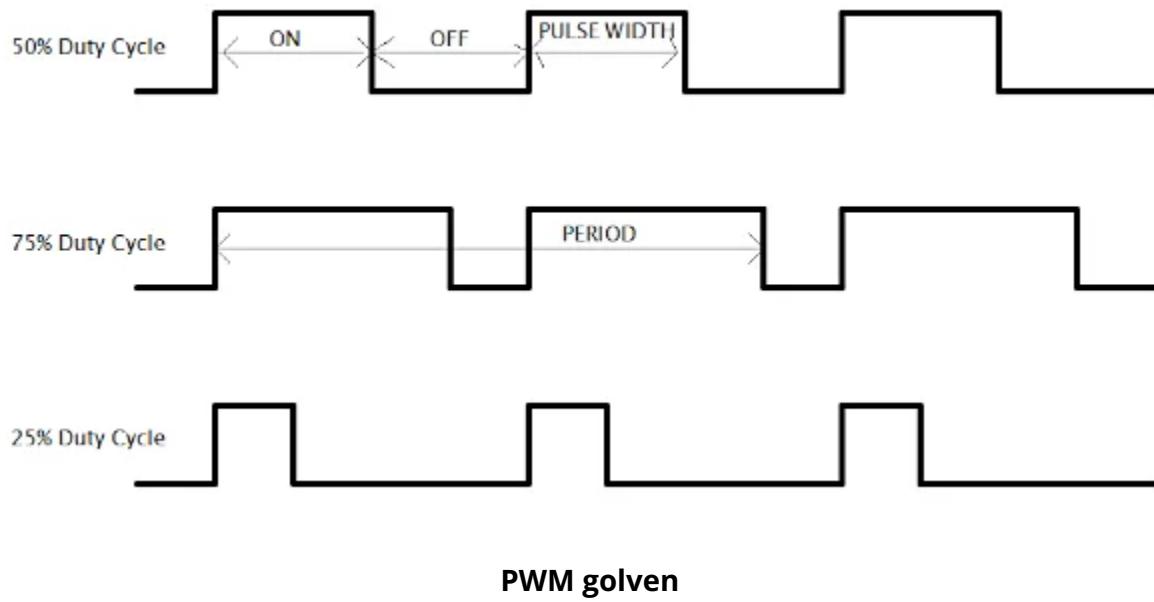
Normaal gesproken moeten wij voor het besturen van de motor van de auto een spanningsbron hebben van minimaal 12 volt, de Raspberry Pi adapter is dus geen optie. De oplossing hiervoor is een batterijpakket dat wij zelf hebben gemaakt met vier Samsung 18650 lithium batteries die wij in serie aansluiten, voor een spanning van ongeveer 12V en een capaciteit van 10.000 mAh, waarmee wij de auto voor meer dan 24 uur kunnen laten werken. Maar er ontstaat dan een ander probleem. Wij hebben geen 5V spanning voor de Lidar-sensor en de Raspberry Pi. Daarom hebben wij besloten om een spanningsregelaar module gebruiken die een ingangsspanning van 9V tot 35V kan omzetten in een uitgangsspanning van 5V en die sterk genoeg is voor hoog stroomverbruik. Deze module is de [Joy-it SBC-Buck02](#).



Joy-it SBC-Buck02

### 4.3.3 Hoe geven wij voltages met de Raspberry Pi door aan de motor en servo van de auto?

Om de snelheid van de auto aan te kunnen passen, moeten wij verschillende voltages doorgeven aan de motor en servo. Maar helaas hebben wij een probleem. Raspberry Pi is, zoals elke computer, een digitaal apparaat en kan dus niet zomaar verschillende voltages outputen en sturen. Daarom moeten wij een techniek gebruiken die het mogelijk maakt om van een vaste voltage, verschillende voltages te kunnen sturen. PWM of [Pulse-width modulation](#), is een modulatietechniek waarbij in een vaste frequentie pulsen worden uitgezonden waarvan de breedte gevarieerd wordt. PWM wordt veel gebruikt als vorm van elektrische voeding.



Doordat de Raspberry Pi digitaal is, kan een signaal dat is gestuurd vanaf de GPIO-pinnen alleen hoog (3.3 volts) of laag (0 volts) zijn. PWM techniek maakt het mogelijk dat we meerdere voltages kunnen sturen en de Raspberry Pi is daarvoor meer dan geschikt. Als wij bijvoorbeeld een voltage van 0.33 volt willen sturen, 10% van 3.3 volt, kunnen wij 10% van alle signalen hoog sturen en de overige 90% laag sturen. Dit is Pulse Width Modulation (PWM). In de python code gebruiken wij de bekende module [gpiozero](#) en in de C++ code gebruiken wij de library [pigpio](#).

```

# Moving Robot
def MoveForward(self, speed):
    self.MovingSpeed = Clamp(speed, 0.0, 1.0)
    self.Motor.forward(self.MovingSpeed)

def MoveBackward(self, speed):
    self.MovingSpeed = Clamp(speed, 0.0, 1.0)
    self.Motor.backward(self.MovingSpeed)

def Stop(self):
    self.Motor.stop()
    self.MovingSpeed = 0

# Giving Directions
def TurnLeft(self):
    self.Steering.min()

def TurnAhead(self):
    self.Steering.mid()

def TurnRight(self):
    self.Steering.max()

# 0.0 to 180.0 (full left : full right) -> mapped to -90.0 : 90.0
def TurnDegree(self, angle):
    self.Steering.angle = angle = Clamp(angle - 90.0, -90.0, 90.0)
    self.MovingHeading = math.radians(angle)

# 0.0 to pi (full left : full right)
def TurnRadian(self, angle):
    return self.TurnDegree(math.degrees(angle))

```

## Functies om de auto te besturen

#### **4.3.4 Hoe sluiten wij de Lidar-sensor aan op de Raspberry Pi?**

Voor het aansluiten van de lidar-sensor op de Raspberry Pi en het verzenden van datapunten zijn er meerdere mogelijkheden. De makkelijkste en meest betrouwbare manier is met gebruik van USB. De lidar-sensor komt met een module die een USB-C en een USB-Micro B heeft. De Raspberry Pi 4B heeft vier USB-poorten, waarop wij de sensor makkelijk kunnen aansluiten. De stroom van de sensor gaat ook over de USB-kabel.

De andere manier is over de UART-poort die beschikbaar is via de GPIO-pinnen op de Lidar-sensor zelf, die wij dan met een paar draden moeten verbinden met de GPIO-pinnen van de Raspberry Pi. Het nadeel van deze methode is dat de rommel groter en de communicatie minder stabiel wordt in vergelijking met een USB-connectie.



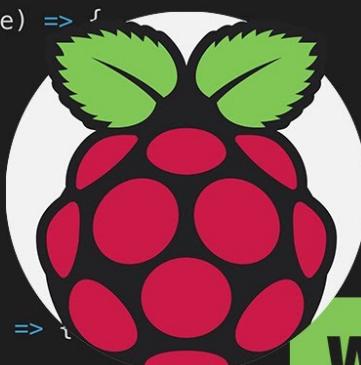
**Eerste versie**

### 4.3.5 Hoe kunnen wij op afstand met de auto communiceren?

Het communiceren met computer kunnen wij doen door gebruik te maken van het bekende protocol SSH. Het [Secure Shell](#) is een cryptografisch netwerkprotocol voor het veilig uitvoeren van netwerkdiensten via een onbeveiligd netwerk. De meest opvallende toepassingen zijn inloggen op afstand en de uitvoering van commands in een terminal van de andere computer op hetzelfde netwerk.

Hiervoor moet de Raspberry Pi wel verbonden zijn met hetzelfde netwerk als de computer die wil communiceren. In onze huis gebruiken wij ons eigen WiFi netwerk en stellen we de Raspberry Pi in om automatisch verbinding te maken met deze WiFi. Op school maken wij met onze telefoon een WiFi netwerk met dezelfde naam en wachtwoord, zodat de Raspberry Pi automatisch verbinding met dit netwerk kan maken. Hierdoor kunnen wij de auto makkelijk op afstand bedienen, alle gegevens zien en zelfs programmeren.

```
1 const Gpio = require('onoff').Gpio;
2 const led = new Gpio(17, 'out');
3 const button = new Gpio(4, 'in', 'both');
4
5 button.watch((err, value) => {
6   if (err) {
7     throw err;
8   }
9
10  led.writeSync(value);
11 });
12
13 process.on('SIGINT', () => {
14   led.unexport();
15   button.unexport();
16 });
17 |
```



**WIFI & SSH**

**SSH protocol**

---

### **4.3.6 Welke programmeertalen gebruiken wij op de Raspberry Pi?**

Het programma van onze robot hebben wij voornamelijk in C++ en Python geschreven. Eerst hadden wij C++ gekozen, omdat wij dachten dat de performance van Raspberry Pi niet genoeg zou zijn als wij het voor de A.I. zouden gebruiken en veel functionaliteiten zouden toevoegen. Maar het blijkt dat de Raspberry Pi wel meer dan genoeg vermogen heeft. Aan de andere kant, om de A.I. beter te kunnen integreren met de rest van de broncode, hebben wij ook een versie in Python geschreven. Met behulp van Python waren we in staat om veel nieuwe functionaliteiten toe te voegen, zoals de auto laten rijden met een PS4 en PS5 controller of via de website die we hebben gemaakt. Deze website maakt het mogelijk om richting te geven aan de auto en de snelheid aan te passen. Verder is er een live feed beschikbaar op de website. Hierbij wordt er realtime een video van de omgeving van de auto gestreamd en uitgezonden.

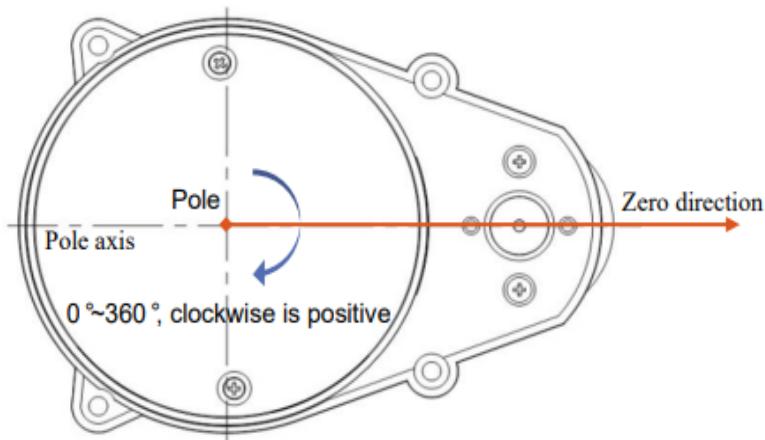
In tegenstelling tot wat we hadden verwacht, blijkt de performance van de Python code niet veel minder te zijn dan die van de C++ versie. Dit komt doordat de kernfunctionaliteit van de helper library YDLidar SDK die wij gebruiken om de lidar sensor te besturen, grotendeels in de taal C++ geschreven is. De python API van deze library gebruikt ook de interface van de snellere C++ code, waardoor de performance hoog blijft. Python heeft ook een ander voordeel, de code hoeft namelijk niet gecompileerd te worden. Daardoor wordt de ontwikkeling van het programma veel makkelijker en sneller voor ons. Daarnaast kunnen we de code sneller testen en debuggen.

Andere reden is dat wij het gewoon leuk vonden om tegelijkertijd in twee verschillende talen te programmeren en het was een goede mogelijkheid om onze kennis van Python te vergroten. Op onze github repository staat het project beschikbaar voor iedereen om het te downloaden. De link is aan het einde van het pws te vinden.

### 4.3.7 Hoe krijgen wij afstandspunten van de Lidar-sensor en geven wij deze door aan de kunstmatige intelligentie?

Per scan van de lidar sensor, krijgen wij ongeveer 750 datapunten die informatie geven over de afstand, hoek en intensiteit van de omgeving. Echter hebben wij de A.I. getraind met een input van 5 richtingen. Deze 750 datapunten moeten wij dus versimpelen en aan de A.I. geven om het te analyseren en de juiste richting te vinden.

YDLidar X4 heeft een bereik van 0.12 tot 10 meter en gebruikt radialen als de eenheid van de hoek. Om onze code efficiënter te maken, gebruiken wij dus overal zo veel mogelijk radialen in plaats van graden. Elke hoek die de sensor aangeeft, komt overeen met een richting in een cirkel en heeft een bereik van  $-\pi$  tot  $\pi$  radialen oftewel  $-180^\circ$  tot  $180^\circ$ . De sensor geeft dus de hoek die groter is dan  $\pi$  radialen negatief aan.



De sensor staat  $180^\circ$  graden tegen de klok in zoals in de bovenstaande afbeelding is weergegeven op de auto. De hoek die overeenkomt met zeven richtingen in de eenheidscirkel wordt als volgt gedefinieerd en in een python constant opgeslagen:

```
HEADING_LEFT = math.pi * 0.0
HEADING_HIGHERLEFT = math.pi * 0.2
HEADING_TOPLEFT = math.pi * 0.4
HEADING_AHEAD = math.pi * 0.5
HEADING_TOPRIGHT = math.pi * 0.6
HEADING_HIGHERRIGHT = math.pi * 0.8
HEADING_RIGHT = math.pi * 1.0
```

#### 4.3.8 Hoe worden de lidar sensor, de camera en de console controller ingesteld?

```
# Setup YDLidar
self.log('Initializing YDLidar SDK')
self.lidar = ydlidar.CYdLidar()
self.lidar.setlidaropt(ydlidar.LidarPropSerialBaudrate, 128000)
self.lidar.setlidaropt(ydlidar.LidarPropLidarType,
ydlidar.TYPE_TRIANGLE)
self.lidar.setlidaropt(ydlidar.LidarPropDeviceType,
ydlidar.YDLIDAR_TYPE_SERIAL)
self.lidar.setlidaropt(ydlidar.LidarPropIgnoreArray, '')
self.lidar.setlidaropt(ydlidar.LidarPropSupportMotorDtrCtrl,
True)
self.lidar.setlidaropt(ydlidar.LidarPropFixedResolution, True)
self.lidar.setlidaropt(ydlidar.LidarPropSingleChannel, True)
self.lidar.setlidaropt(ydlidar.LidarPropAutoReconnect, True)
self.lidar.setlidaropt(ydlidar.LidarPropIntenstiy, False)
self.lidar.setlidaropt(ydlidar.LidarPropReversion, False)
self.lidar.setlidaropt(ydlidar.LidarPropInverted, False)
self.lidar.setlidaropt(ydlidar.LidarPropAbnormalCheckCount, 4)
self.lidar.setlidaropt(ydlidar.LidarPropScanFrequency, 12.0)
self.lidar.setlidaropt(ydlidar.LidarPropIntenstiyBit, 0)
self.lidar.setlidaropt(ydlidar.LidarPropSampleRate, 5)
self.lidar.setlidaropt(ydlidar.LidarPropMaxRange, 10.0)
self.lidar.setlidaropt(ydlidar.LidarPropMinRange, 0.12)
self.lidar.setlidaropt(ydlidar.LidarPropMaxAngle, 180.0)
self.lidar.setlidaropt(ydlidar.LidarPropMinAngle, -180.0)
self.scan = ydlidar.LaserScan()

# Setup camera
self.log(f'Initializing Camera with resolution {CAMERA_WIDTH} * {CAMERA_HEIGHT}')
KillProcesses('lsof -t /dev/video0')
self.camera = Picamera2()
self.camera_config = self.camera.create_still_configuration(main={'size': (CAMERA_WIDTH, CAMERA_HEIGHT)})
self.camera.configure(self.camera_config)
self.camera.start()
self.InputImage = self.camera.capture_array()
```

```
self.ImageNodes = []
self.ImageIndex = 0

# Connect YDLidar
if not self.ConnectLidar():
    sys.exit()

# Setup Console Controller
Thread(target=self.ConnectController, daemon=True).start()

# Search for Console Controller
def ConnectController(self):
    # Iterate through all joystick interfaces
    address = f'{CONTROLLER_ADDRESS}{0}'
    for idx in range(0, 8):
        if os.path.exists(f'{CONTROLLER_ADDRESS}{idx}'):
            address = f'{CONTROLLER_ADDRESS}{idx}'
            break

    # Controller found
    self.controller = ConsoleController(interface=address,
connecting_using_ds4drv=False, callback=self)
    Thread(target=self.controller.listen, args=[sys.maxsize],
daemon=True).start()

    time.sleep(0.25)
    if self.controller.is_connected:
        self.log(f'Connected to controller at {address}')
        return True

    self.log('No console controller could be found')
    return False
```

#### 4.3.9 Hoe kunnen wij verbinding maken met de Lidar en de controller?

```
# Search for YDLidar devices

def ConnectLidar(self):

    portList = ydlidar.lidarPortList().items()
    if len(portList) == 0:
        self.log('No YDLidar device could be found')
        return False

    # Iterate through all YDLidar devices
    for version, port in portList:
        # YDLidar Found
        self.log(f'YDLidar device has been found - Port: {port}, Version: {version}')

        # Try to Initialize YDLidar SDK
        self.lidar.setlidaropt(ydlidar.LidarPropSerialPort, port)
        if self.lidar.initialize() and self.lidar.turnOn():
            return True
        else:
            self.log(f'Failed to connect to YDLidar at {port}')
            return False

# Search for Console Controller
def ConnectController(self):
    # Iterate through all joystick interfaces
    address = f'{CONTROLLER_ADDRESS}{0}'
    for idx in range(0, 8):
        if os.path.exists(f'{CONTROLLER_ADDRESS}{idx}'):
            address = f'{CONTROLLER_ADDRESS}{idx}'
            break

    # Controller found
    self.controller = ConsoleController(interface=address,
                                         connecting_using_ds4drv=False, callback=self)
    Thread(target=self.controller.listen, args=[sys.maxsize],
           daemon=True).start()

    time.sleep(0.25)
```

```
if self.controller.is_connected:  
    self.log(f'Connected to controller at {address}')  
    return True  
  
self.log('No console controller could be found')  
return False
```

#### 4.3.10 Hoe controleren wij de auto en geven wij commando's?

Wij hebben twee verschillende manieren bedacht om de auto te controleren en commando's geven, zoals controleren met een PS4 of PS5 controller of met de website van het project. De Raspberry Pi begint met een HTTP server. Alle gebruikers die op hetzelfde netwerk als dat van de Raspberry Pi zitten kunnen de server in hun browser openen.

```
self.webserver = HTTPSserver((GetIpAddress(), 8080),  
functools.partial(HTTPHelper, callback=self))  
Process(target=self.webserver.serve_forever, daemon=True).start()  
Process(target=os.system, args=[f'python3 -m http.server -b  
{GetIpAddress()}'], daemon=True).start()  
self.log(f'WebServer started at {GetIpAddress():{self.webserver.server_address[1]}} and  
{GetIpAddress():8000}' )
```

De onderstaande website wordt gehost op de server die door de Raspberry Pi wordt gemaakt. Het adres van deze pagina is als volgt: "IP address van de Pi:8080".

### 4.3.11 Hoe wordt de TensorFlow Lite library ingesteld?

Elk model dat in de TensorFlow wordt gemaakt heeft een input en output shape. Dit beschrijft hoe de data eruit zag en is essentieel voor het juist functioneren van de AI.

De onderstaande functie wordt tijdens de initialisatie opgeroepen.

```
# Setup TensorFlow
def SetTensorFlow(self):
    if USE_CAMERA:
        self.tfInterpreter = tf.Interpreter(AI_MODEL_CAMERA_FILE)
    else:
        self.tfInterpreter = tf.Interpreter(AI_MODEL_LIDAR_FILE)
    self.tfInterpreter.allocate_tensors()

    # Get input and output tensors.
    self.input_details = self.tfInterpreter.get_input_details()
    self.output_details = self.tfInterpreter.get_output_details()
    self.input_shape = self.input_details[0]['shape']
```

### 4.3.12 Hoe wordt het beeld van de camera omgezet naar een format dat verwerkt kan worden door de A.I.?

De A.I. kan normaal gesproken niet met een onbewerkte foto werken. Het beeld moet omgezet worden naar een format dat herkenbaar is en verwerkt kan worden. Daarvoor hebben wij een functie gemaakt.

```
def ProcessImage(self, image):
    image = cv2.resize(image, (250, 250))
    height, _, _ = image.shape
    image = image[int(height/2):, :, :]
    image = cv2.GaussianBlur(image, (5, 5), 0)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = cv2.Canny(image, 50, 100)
    image = cv2.resize(image, (250, 250))
    image = image.reshape(250, 250, 1)
    image = image / 255 # normalizing
    return image
```

---

### 4.3.13 Hoe wordt de data van de Lidar gefilterd op basis van de gegeven hoek?

De lidar sensor draait voortdurend 360° en geeft van elke hoek twee outputs. Om de afstand van een bepaalde hoek te vinden moeten we de data van de lidar filteren en alleen de data van de gewenste hoek eruit halen. Dit wordt in de onderstaande functie gedaan.

```
# Filter scans based on angle, returns a distance
def FilterScansAngle(self, angle):
    results = [Clamp(scan.range, 0.0, 1.0) for scan in self.scan.points
if math.isclose(scan.angle, angle, rel_tol=DEFAULT_APERTURE)]
    if len(results):
        return numpy.mean(results)
    return 0.0

# Filter scans based on maximum distance, returns a scan
def FilterScansMaxDistance(self):
    results = [scan for scan in self.scan.points if HEADING_LEFT <=
scan.angle <= HEADING_RIGHT]
    return max(results, key=lambda scan: scan.range)
```

### 4.3.14 Hoe verzamelen we data om de Artificial Intelligence te trainen?

Om de A.I makkelijker te kunnen verzamelen hebben wij een functie gemaakt. Als de gebruiker de knop 'X' op de controller ingedrukt houdt, wordt deze functie geroepen en de auto begint data te verzamelen van zijn omgeving. Als de gebruiker deze knop loslaat, stopt de auto met het verzamelen van data en slaat die data op in verschillende bestanden. Om toegang tot deze bestanden makkelijker te maken, hebben wij een webserver gemaakt. De gebruiker heeft dan toegang tot alle bestanden in de auto en kan ze downloaden. Op de volgende bladzijde is deze functie terug te vinden.

```

    if self.RecordNodes:
        if USE_CAMERA:

            self.camera.capture_file(f'{AI_FOLDER_IMAGES}/{self.ImageIndex}.jpg')

            self.ImageNodes.append(f'Images/img{self.ImageIndex}.jpg,
{self.MovingHeading}, {self.MovingSpeed}')
                self.ImageIndex += 1
        else:
            self.DistanceNodes.append([
                self.FilterScansAngle(HEADING_LEFT),
                self.FilterScansAngle(HEADING_HIGHERLEFT),
                self.FilterScansAngle(HEADING_TOPLEFT),
                self.FilterScansAngle(HEADING_TOPRIGHT),
                self.FilterScansAngle(HEADING_HIGHERRIGHT),
                self.FilterScansAngle(HEADING_RIGHT),
            ])
            self.InputNodes.append([self.MovingSpeed,
self.MovingHeading / math.pi])

```

### 4.3.15 Hoe laten we de Artificial Intelligence de auto besturen?

Om de juiste koers te kiezen en de auto in die richting te laten bewegen, moeten alle datapunten die de Lidar-sensor verzendt gegeven worden aan het getrainde neurale netwerk of TensorFlow. De weights, biases en de data worden in een los bestand opgeslagen. Dit maakt het makkelijker als wij tijdens het testen de werking van de neural networks willen veranderen. Eerst laden wij het opgeslagen neurale netwerk op en creëren wij op basis daarvan een nieuw model. We geven in heel korte tijd de afstand van de auto tot zijn omgeving aan het A.I. en vervolgens berekent het A.I. voor ons de steering angle en / of de auto gas moet geven. Dit krijgen wij in een node array terug. Op de volgende bladzijde is deze functie terug te vinden.

```

if self.FollowAI:
    if USE_CAMERA:
        self.InputImage = self.camera.capture_array()
        self.InputImage = self.ProcessImage(self.InputImage)

            self.InputImage = numpy.array([self.InputImage],
dtype=numpy.float32)

self.tfInterpreter.set_tensor(self.input_details[0]['index'],
self.InputImage)
    self.tfInterpreter.invoke()
    tensor      =
self.tfInterpreter.get_tensor(self.output_details[0]['index'])
    node = tensor[0]
    # print(node)

        heading = MapRange(node[0], -1.0, 1.0, HEADING_LEFT,
HEADING_RIGHT)
        self.TurnRadian(heading)
        self.MoveForward(self.MovingSpeed)

else:
    distanceNodes = numpy.array(
    [
        [
            self.FilterScansAngle(HEADING_LEFT),
            self.FilterScansAngle(HEADING_HIGHERLEFT),
            self.FilterScansAngle(HEADING_TOPLEFT),
            self.FilterScansAngle(HEADING_TOPRIGHT),
            self.FilterScansAngle(HEADING_HIGHERRIGHT),
            self.FilterScansAngle(HEADING_RIGHT),
        ]
    ],
    dtype=numpy.float32,
)

self.tfInterpreter.set_tensor(self.input_details[0]['index'],
distanceNodes)
    self.tfInterpreter.invoke()
    tensor      =

```

```
self.tfInterpreter.get_tensor(self.output_details[0]['index'])
    node = tensor[0]
    # print(node)

        heading = MapRange(node[1], -1.0, 1.0, HEADING_LEFT,
HEADING_RIGHT)
        self.TurnRadian(heading)
        self.MoveForward(self.MovingSpeed)
```

### Controle van de auto door A.I.

---

## **4.4 Werkelijkheid**

Naast de simulatie hebben wij natuurlijk ook de werkelijkheid. Het doel van dit profielwerkstuk is uiteindelijk om de auto echt te laten rijden en om de simulatie naar realiteit te brengen. Zo kunnen wij onze hoofdvraag beter beantwoorden.

### **4.4.1 Wat willen wij onderzoeken met ons experiment?**

Het doel van het experiment is om te onderzoeken hoe betrouwbaar een A.I. is met het besturen van een auto. We gaan dit testen door de A.I. rond te laten rijden in een parcours en te observeren hoeveel fouten de A.I. hierbij maakt. Ook gaan we analyseren hoe gevaarlijk deze fouten werkelijk zijn. Verder gaan we testen hoe de A.I. handelt in een paar speciale scenario's die misschien in het echt voor zouden kunnen komen. In al deze testen is de snelheid en de rijstijl ook belangrijk. Het is niet goed als de auto heel traag gaat rijden of onverwachte bochten maakt zonder reden.

Een minder belangrijk doel van het experiment is erachter komen hoe snel A.I.'s leren en hoeveel potentie een A.I. heeft. Of het aan de gang krijgen van de A.I. heel moeizaam verloopt, of dat de magie van de A.I. zelf al het werk doet. We willen dus vooral ervaring krijgen met het gebruiken van AI.

---

#### **4.4.2 Hoe gaan wij het onderzoek uitvoeren?**

Om met het onderzoek te kunnen beginnen moet de A.I. eerst getraind worden. Onze A.I. leert op basis van trainingsdata die we eerst nog zelf moeten maken. Dit gaan we doen door eerst een parcours te maken waar de auto in rond kan rijden. We hopen hiervoor een lokaal te kunnen gebruiken en met tafels een parcours op te zetten, maar als dat niet mogelijk is gaan we een van boeken parcours maken in ons eigen huis. Vervolgens gaan we zelf de auto besturen terwijl de auto rondjes rijdt in het parcours. We gaan hierbij alle inputs en outputs opslaan en gebruiken als trainingsdata voor de A.I. Hierna laten we de A.I. trainen op deze data.

Nadat de A.I. klaar is met trainen gaan we meerdere tests mee uitvoeren. Ten eerste gaan we testen of de A.I. in het parcours rond kan rijden met als doel dat hij niet tegen de muren aan rijdt of voorkomen dat hij helemaal niet wil rijden. We gaan hierbij ook kijken hoe snel hij gaat en of zijn rijstijl lijkt op die van de trainingsdata. Verder gaan we testen hoe de A.I. reageert in speciale scenario's. Hierbij gaan we bijvoorbeeld een bewegend object voor de auto zetten en kijken of hij het kan ontwijken. In het echt zou dat een auto of een fietser kunnen voorstellen. Een ander speciaal scenario is dat er een vogel of blaadje voor een van de sensoren langs vliegt. We gaan testen hoe de A.I. hierop reageert door onze hand kort voor de sensoren te houden. Bij al deze testen gaan we noteren hoe de A.I. reageert en analyseert of dat wel veilig is.

---

## H5 Conclusie

Het is belangrijk om te kijken naar vier verschillende deelonderwerpen die met elkaar het hele PWS omvatten. Deze deelonderwerpen zijn de deelvragen van dit PWS. Hierdoor kunnen wij een antwoord geven op onze hoofdvraag.

### 5.1 Simulatie

Onze simulatie is het meest geschikt voor het trainen van een neuraal netwerk binnen een korte tijd. Het opzetten van de simulatie is makkelijk en duurt niet lang. We kunnen heel veel verschillende omstandigheden creëren waaronder onze A.I. zich moet aanpassen. Het komt heel dicht bij de realiteit omdat we van een bekende gaming engine, Unity, gebruik hebben gemaakt. Er werken namelijk weerstandskracht, draagkracht, maximumsnelheid en nog veel meer natuurkundige simulaties op onze auto.

### 5.2 Artificial Intelligence

We hebben onze getrainde A.I. in verschillende onbekende routes in de simulatie laten rijden. Het A.I. kon heel goed door deze routes heen maar het kan nog beter functioneren als we het getrainde A.I. eerst met die onbekende routes hadden getraind. Met lidar was het neuraal netwerk nauwkeuriger in het voorspellen van de juiste stuurhoek en hoeveelheid gas. Een manier om camera en lidar met elkaar te combineren zou dus een beter resultaat moeten opleveren.

## H6 Discussie

Vaak tijdens het onderzoek doen naar onze hoofdvraag zijn we erachter gekomen dat we niet met het correcte bezig waren, of iets te lastig aan het doen waren. Met deze fouten hebben we wel een hoop geleerd over hoe AI's werken. Zo hebben we eerst geprobeerd zelf een AI te schrijven, maar dit bleek lastiger te zijn dan wij dachten. De A.I. leerde erg traag en we zouden de auto honderden keren moeten hebben laten oefenen met de echte auto voordat het zou kunnen werken. Dit kost veel te veel tijd. Ook lukte het niet om de A.I. te trainen in de simulatie en in de realiteit te laten rijden, daarom hebben wij later besloten een A.I. uit een library te gebruiken. Deze leert op basis van een voorbeeld dat we de A.I. geven in plaats van zelf alles uit te gaan testen.

---

Ook bedachten we later in ons pws dat een A.I. niet genoeg had aan de sensoren die we de auto hadden gegeven, wat heeft het voor zin als de auto kan zien dat er iets voor hem zit als hij niet weet wat het is? Daarom hebben wij later gebruikgemaakt van een camera. Met een camera wordt het mogelijk om objecten rondom de auto te herkennen en op tijd deze te vermijden. Met een lidar sensor was dit helaas niet mogelijk, want met een lidar sensor kan alleen maar de afstand tussen de auto en zijn omgeving bepaald worden. Met een camera kunnen ook, bijvoorbeeld, verschillende verkeersborden herkend worden en de A.I. kan hierop reageren.

Een vervolgonderzoek zou ook kunnen zijn om de input data van lidar en camera samen te voegen in één neuraal netwerk. Het betekent dat het neuraal netwerk tegelijkertijd de input data van de lidar sensor en afbeeldingen van een camera binnen krijgt. Het nut hiervan is de nauwkeurigheid van onze AI. De A.I. kan beter in onze simulatie met een lidar sensor rijden, want een lidar sensor levert een meer nauwkeurige informatie over de afstanden op. Maar in werkelijkheid wordt deze data noisy omdat de niet alleen de muren maar ook andere echte objecten worden gemeten door de lidar sensor. Met een camera is het echter mogelijk om lijnbanen te herkennen, maar als de auto te dicht bij een object komt die buiten het zichtveld van de camera ligt, zou de auto crashen tegen dit object. Een lidar sensor kan deze op tijd meten en het neuraal netwerk kan op tijd reageren.

In een vervolgonderzoek zouden we ook de veiligheid en meerdere eigenschappen zoals de reactiesnelheid van de AI in de realiteit willen testen, maar de auto ging kapot en kon niet op tijd gerepareerd worden voor de einddatum. Dit heeft ervoor gezorgd dat een deel van ons onderzoek niet uitgevoerd kon worden. Wel hebben we de simulatie werkend gekregen, met een auto die betrouwbaar rond kan rijden in een parcours.

Achteraf gezien zouden we in een vervolgonderzoek wel kiezen voor dezelfde hardware, ondanks dat de raspberry pi kapot was gegaan. De robot heeft gewerkt, maar is kapot gegaan bij het toevoegen van de camera. Ook hebben we een goede keuze gemaakt bij welke programmeertaal we gingen gebruiken. We hebben er geen problemen mee gehad. En de ontwikkeling van het AI en de raspberry pi ging wat ons betreft makkelijk door.

---

## H7 Nawoord

### 7.1 Broncode

De broncode van dit PWS is voor publiek beschikbaar in de repository van dit project op github en wordt daar onderhouden. Een uitgebreide uitleg over de broncode is te vinden in het README.md bestand. Hierin staat ook hoe men de software van dit project makkelijk kan downloaden en installeren.

<https://github.com/SadraShameli/ProjectAI>

# Project A.I.

The purpose of this project is to create an autonomous self-driving robot, which is able to follow a course and avoid obstacles, all on its own. Its inspiration originates from being a big fan of Elon Musk, Tesla and its technology. This project is mainly created for PWS (profielwerkstuk) using our own specific hardware, but it can be replicated by the user to work on their hardware as well.

You can visit my social profiles using the following links: [YouTube](#) | [Linkedin](#) | [Instagram](#)



## Features

- Fully autonomous driving without any input from the user
- Ability to manually control using a PS4 or PS5 controller
- Ability to manually control using the website created on the local WiFi network
- Displaying a 2D model of the surroundings on the website
- Making use of the well-proven machine learning library [TensorFlow](#)
- Using threads and thread pooling for every core functionality
- Source code written in both Python and C++ (currently under development)

## Project A.I. Github Repository

---

## Appendix

### A.1 Logboek

Taak	Jelle	Sadra	Vere	Parsa	Datum	Som
PWS - presentaties bijwonen	1	1	1	0		3
PWS - presentatie (algemeen)	10	10	10	10		40
PWS - overleg met begeleider	6	6	6	3		21
Onderwerp uitkiezen	5	5	5	5		20
Plan van aanpak + planning	5	5	5	0		15
Hoofdvraag formuleren	2	2	2	0		6
Deelvragen formuleren	2	2	2	0		6
PWS overleg met begeleider	0,5	0,5	0,5	0		1,5
Plan van aanpak verbeteren	3	3	3	0		9
Planning hoofdstukken	3	3	3	0		9
Begin hoofdstukken	1,5	1,5	1,5	0		4,5
Robot in elkaar zetten, programmeren	0	48	0	0		48
Python, C# in unity leren	0	0	0	20		20
Ontwerpen simulatie	0	0	0	25		25
Python leren	5	0	0	0		5
Ontwerpen neuraal netwerk	25	0	0	25		50
H2 Inleiding schrijven	0	0	3	0		3
Hoofdstukken schrijven	3	3	10	0		16
Spelling check	0	0	10	0		10
H1 Voorwoord schrijven	2	0	0	0		2
Deelonderwerp neuraal netwerk schrijven	5	0	0	1		6
Deelonderwerp de werkelijkheid schrijven	4	0	0	0		4
Discussie schrijven	2	0	0	0		2
Deelonderwerp simulatie	0	0	0	5		5
Bijhouden van het logboek	2	2	2	2		8
Conclusie schrijven	0	0	3	0		3

---

H3 theoretische achtergrond schrijven (behalve neuraal netwerk)	0	20	0	0		20
H4 deelvragen	0	31	0	0		20
Presentatie	0	0	10	0		6
Totaal	87	143	77	96		382

---

## A.2 Fotos



**Onderdelen van de auto**



Het opbouwen van de chassis

---

### A.3 Onderdelen, Kosten

Helaas was ons PWS niet gratis. We hebben best wel veel geld besteed aan ons PWS. Dus iemand die ons onderzoek wil nadoen heeft de volgende producten / onderdelen nodig. Prijzen zijn correct op het moment van schrijven.

Product	Prijs	Hoeveelheid
Raspberry Pi 4B - Kit	€ 67.95	1
Raspberry Pi 4B - Power Supply	€ 8.95	1
Raspberry Pi Camera V2 - 8MP	€ 28.50	1
Raspberry Pi Heatsink case with fan Black	€ 16.10	1
YDLIDAR X4 Lidar – 360° Laser Range Scanner	€ 74.95	1
Transcend 32GB Micro SD	€ 12.36	1
Micro HDMI cable 1.8 meters	€ 7.56	1
4WD RC Smart Car Chassis met Servo	€ 49.95	1
Joy-it SBC-Buck02 Voltage regulator	€ 12.49	1
Joy-IT L298N Motor driver	€ 7.99	1
40 Premium Jumper Wires 20cm F/F	€ 4.45	2
<b>Totaal</b>	<b>€ 291.25</b>	

---

## Bronnenlijst

Understanding Error Backpropagation. (Oct 23, 2020). Geraadpleegd van  
<https://towardsdatascience.com/error-backpropagation-5394d33ff49b>

Artificial Intelligence, Enough of the hype! What is it?. (May 17, 2019). Geraadpleegd van  
<https://community.hpe.com/t5/hpe-blog-uk-ireland/artificial-intelligence-enough-of-the-hype-what-is-it/ba-p/7046672>

AI IN AUTOMOTIVE: A NEW EDGE OF THE AUTOMOTIVE INDUSTRY. (June 18, 2021).  
Geraadpleegd van  
<https://nix-united.com/blog/ai-in-automotive-a-new-edge-of-the-automotive-industry>

Autonomoos rijden in stroomversnelling door sensortechniek. (2017). Geraadpleegd van  
<https://e.sentech.nl/nieuws/autonomoos-rijden-in-stroomversnelling-door-sensortechniek>

LiDAR vs. RADAR. (April 24, 2018). Geraadpleegd van  
<https://www.fierceelectronics.com/components/lidar-vs-radar>

What, Why and Which?? Activation Functions. (April 14, 2019). Geraadpleegd van  
<https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441>

Map number to range. (April 5, 2021). Geraadpleegd van  
<https://www.30secondsofcode.org/python/s/num-to-range>

How to SSH into a Raspberry Pi. (May 13, 2019). Geraadpleegd van  
<https://itsfoss.com/ssh-into-raspberry/>

---

## **Libraries**

### **Auto**

Fillion-Robin, J. (November 17, 2022). CMake. Geraadpleegd van  
<https://github.com/Kitware/CMake>

Beazley, D. (October 28, 2022) swig. Geraadpleegd van  
<https://swig.org/svn.html>

Houghton, A. (May 31, 2021). netifaces. Geraadpleegd van  
<https://github.com/al45tair/netifaces>

YDLidar. (March 14, 2020). YDLidar-SDK . Geraadpleegd van  
<https://github.com/YDLIDAR/YDLidar-SDK>

Jones, D. (September 30, 2020). pigpio. Geraadpleegd van  
<http://abyz.me.uk/rpi/pigpio/python.html>

Nuttall, B. (March 18, 2021) Geraadpleegd van  
<https://github.com/gpiodriver/gpiodriver>

Spirin, A. (April 3, 2021). pyPS4Controller. Geraadpleegd van  
<https://github.com/ArturSpirin/pyPS4Controller>

### **Kunstmatige Intelligentie**

Google. (November 9, 2015). TensorFlow. Geraadpleegd van  
<https://github.com/tensorflow/tensorflow>

Google. (March 6, 2019). TensorFlow lite. Geraadpleegd van  
<https://www.tensorflow.org/lite>

Intel. (June 2000). OpenCV. Geraadpleegd van  
<https://opencv.org>

---

## **Simulatie**

Unity Technologies. (June 2005). Unity Engine. Geraadpleegd van  
<https://unity.com>