Is het mogelijk om een PID controller te vervangen door een AI?

Mitchel Reints - 1040953@hr.nl
Thomas van Ommeren - 1033239@hr.nl
Hogeschool Rotterdam - Elektrotechniek - PEE51

21 juni 2025

Samenvatting
In dit onderzoek wordt onderzocht of een neuraal netwerk het gedrag van een klassieke PID-regelaar kan nabootsen of zelfs verbeteren binnen een systeem waarin een pingpongbal op een constante hoogte wordt gehouden door middel van een ventilator. De huidige opstelling op de Hogeschool Rotterdam gebruikt een microcontroller en een PID-regelaar om de hoogte van de bal te regelen. Het afstemmen van deze regelaar is echter gevoelig voor omgevingsveranderingen en vereist handmatige tuning. Met de opkomst van machine learning rijst de vraag of een zelflerend model betere prestaties kan leveren. In dit project is een neuraal netwerk ontworpen, getraind en getest in een gesimuleerde omgeving. Het model is geëvalueerd op nauwkeurigheid, stabiliteit en generaliseerbaarheid, en vervolgens vergeleken met de klassieke PID-regelaar. De resultaten tonen aan dat het neurale netwerk het gedrag van de PID-regelaar goed kan benaderen en in sommige gevallen zelfs verbeterde prestaties levert, mits correct getraind. Hoewel de uiteindelijke implementatie op embedded hardware buiten de scope van dit onderzoek valt, is bij de modelkeuze wel rekening gehouden met beperkingen van embedded systemen. Dit onderzoek biedt waardevolle inzichten in het toepassen van kunstmatige intelligentie voor regeltechniek in embedded omgevingen en dient als opstap voor verdere integratie van slimme regelaars in praktische toepassingen.

1 Inleiding

Hogeschool Rotterdam beschikt over de cursus Digitale Systemen (DIS10) over een opstelling waarbij een pingpongbal op een ingestelde hoogte wordt gehouden. Deze opstelling bestaat uit een verticale buis met onderaan een ventilator. Door de luchtdruk van de ventilator wordt de bal omhoog geblazen. Met behulp van een microcontroller wordt de ventilator aangestuurd, zodat de bal op een gewenste hoogte blijft zweven. De regeling van de ventilatorsnelheid gebeurt momenteel met behulp van een PID-regelaar (Proportioneel-Integrerend-Differentieel), waarbij de afwijking tussen de gewenste en werkelijke hoogte continu wordt bijgestuurd.

Hoewel deze opstelling functioneert zoals bedoeld, is het instellen van de PID-parameters vaak een handmatig proces. Het vergt tijd om het systeem goed af te stemmen om het uiteindelijk stabiel te laten reageren op hoogteveranderingen. Daarnaast kunnen veranderingen in de omgeving (zoals temperatuur of luchtvochtigheid) de prestaties van de PID-regelaar beïnvloeden.

De begeleidende docent heeft de vraag gesteld of deze traditionele regelmethode vervangen kan worden door een benadering gebaseerd op machine learning. Machine learning biedt de mogelijkheid om op basis van data te leren hoe het systeem zich moet gedragen. In dit project wordt onderzocht of het mogelijk is om een machine learning model te ontwikkelen dat het gedrag van de PID-regelaar kan nabootsen of zelfs verbeteren.

Dit onderzoek is relevant omdat het laat zien hoe klassieke regeltechniek vervangen kan worden door moderne, zelflerende systeem. Daarnaast sluit het aan bij actuele ontwikkelingen binnen de techniek en biedt het studenten inzicht in zowel embedded systemen als kunstmatige intelligentie. Uiteindelijk is het doel om een demonstratie-opstelling te creëren die gebruikt kan worden tijdens bijvoorbeeld open dagen, om zo op een toegankelijke manier beide technieken te tonen.

Het doel van dit onderzoek is om te onderzoeken of een neuraal netwerk het gedrag van een PID-regelaar kan nabootsen of verbeteren in een systeem waarbij een pingpongbal op een ingestelde hoogte wordt gehouden. Hierbij wordt een model ontwikkeld, getraind en geëvalueerd in een gesimuleerde omgeving. Uiteindelijk moet het model geschikt zijn voor implementatie op een embedded platform, met als doel een stabiele en nauwkeurige hoogtecontrole te realiseren. Het onderzoek richt zich op zowel de technische haalbaarheid als de vergelijking met de bestaande PID-regelaar. we gaan in dit onderzoek niet in op de implementatie van een neuraal netwerk in een embedded omgeving, maar er wordt wel rekening gehouden met dat het uiteindelijk in een embedded omgeving geimplementeerd moet worden. In dit onderzoek wordt de nadruk gelegd op of het mogelijk is om een neuraal netwerk het PID-regelaar gedrag te laten nabootsen en hoe dit zich verhoudt tot de traditionele PID-regelaar.

Voor dit onderzoek wordt er rekening gehouden met het neurale netwerk op de **NXP FRDM-MCXN947 Development board** microcontroller. In het verloop van het onderzoek wordt er rekening gehouden met de beperkingen van deze microcontroller, zoals geheugen- en rekenkrachtbeperkingen. Dit is belangrijk omdat het uiteindelijke doel is om het getrainde model op deze microcontroller te implementeren. De NXP FRDM-MCXN947 Development board is gekozen omdat deze geschikt is voor embedded toepassingen en voldoende mogelijkheden biedt voor het uitvoeren van machine learning taken.

Voor dit onderzoek zijn er een aantal onderzoeksvragen geformuleerd die uiteindelijk de hoofdvraag van dit onderszoek moeten beantwoorden:

1. Hoe werkt een PID-regelaar en hoe wordt deze toegepast in de huidige opstelling?
2. Welke soorten neurale netwerken zijn geschikt voor regeltoepassingen in embedded systemen?
3. Hoe kan trainingsdata verzameld worden om het gedrag van de PID-regelaar na te bootsen?
4. Hoe presteert het getrainde neurale netwerk vergeleken met de traditionele PID-regelaar?
5. Wat zijn de beperkingen van het gebruik van een neuraal netwerk in een embedded omgeving?

In dit rapport wordt eerst de benodigde theorie besproken (hoofdstuk 2), gevolgd door de gebruikte methoden (hoofdstuk 3) waarin we dit onderzoek doen. Daarna worden de resultaten gepresenteerd (hoofdstuk 4) en besproken (hoofdstuk 5). Het rapport sluit af met conclusies en aanbeveling (Hoofdstuk 6).

aan het einden van het onderzoek zullen deze deelvragen worden beantwoord en onderbouwd, zodat de hoofdvraag kan worden beantwoord: *Is het mogelijk om een PID-regelaar te vervangen door een AI?*

2 Theorie

2.1 Werking van een PID-regelaar

Een PID-regelaar (Proportioneel-Integrerend-Differentiërend)[8] [1] is een veelgebruikte terugkoppelingsregelaar in de regeltechniek, die als doel heeft een systeem naar een gewenste waarde (setpoint) te sturen door continu het verschil (de fout) tussen die gewenste waarde en de gemeten systeemoutput te corrigeren. De PID-regelaar bestaat uit drie componenten:

- **Proportioneel (P):** Deze component reageert op de huidige fout. Hoe groter de fout, hoe sterker de aansturing. Dit zorgt voor een directe correctie, maar kan leiden tot een blijvende fout (steady-state error) als deze component alleen wordt gebruikt.
- **Integrerend (I):** Deze component reageert op de opgetelde (geïntegreerde) fout in de tijd. Hierdoor kan de regelaar kleine resterende fouten wegwerken en wordt het systeem naar het exacte setpoint geduwd. Te veel integratie kan echter leiden tot traagheid of instabiliteit.
- **Differentiërend (D):** Deze component reageert op de snelheid waarmee de fout verandert. Dit helpt om snelle veranderingen te dempen en voorkomt overshoot door het systeem te vertragen voordat het het setpoint bereikt. De D-term maakt het systeem reactiever en stabieler bij plotselinge veranderingen.

De regeloutput $u(t)$ van een PID-regelaar wordt meestal als volgt beschreven:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)\, d\tau + K_d \cdot \frac{de(t)}{dt} \qquad (1)$$

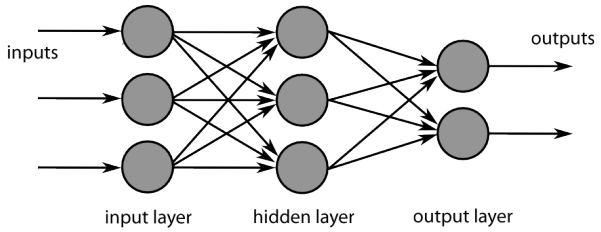
waarbij:

$e(t) = r(t) - y(t)$ (de fout tussen de referentie en gemeten waarde)
$K_p =$ proportionele versterkingsfactor
$K_i =$ integrerende versterkingsfactor
$K_d =$ differentiërende versterkingsfactor

Een goed afgestemde PID-regelaar zorgt voor een snelle respons, minimale overshoot en een stabiele benadering van het setpoint zonder blijvende fout. Hoewel we in dit onderzoek niet in detail ingaan op de afstemming van PID-parameters, is het belangrijk te vermelden dat dit een cruciaal onderdeel is van het gebruik van PID-regelaars in de praktijk. De afstemming wordt meestal handmatig gedaan en dit is meestal een iteratief proces waarbij de waarden van K_p, K_i en K_d worden aangepast op basis van de prestaties van het systeem. Er zijn verschillende methoden voor PID-afstemming, zoals de Zieglers-Nichols-methode, maar deze vallen buiten de scope van dit onderzoek. In dit onderzoek richten we ons op het vervangen van de PID-regelaar door een neuraal netwerk, waarbij we de nadruk leggen op het gedrag van de PID regelaar en de mogelijkheid om dit gedrag te repliceren met een neuraal netwerk.

2.2 Neurale netwerken - basisprincipes

[2] [7]Een neuraal netwerk is een wiskundig model dat is geïnspireerd op de werking van het menselijk brein. Het bestaat uit een verzameling onderling verbonden knopen, ook wel neuronen genoemd, die georganiseerd zijn in lagen: één inputlaag, één of meerdere verborgen lagen en één outputlaag. Elke verbinding tussen neuronen heeft een gewicht dat bepaalt hoe sterk een input wordt doorgegeven. Tijdens het doorlopen van het netwerk wordt elke input vermenigvuldigd met het bijbehorende gewicht, opgeteld met een bias en vervolgens door een activatiefunctie gehaald. Deze activatiefunctie bepaalt of en in welke mate het neuron wordt geactiveerd. Het leerproces van een neuraal netwerk gebeurt aan de hand van trainingsdata. Op basis van de fout tussen de voorspelde output en de werkelijke waarde worden de gewichten aangepast met behulp van een algoritme zoals bijvoorbeeld backpropagation in combinatie met gradient descent. Door herhaaldelijk trainen op veel voorbeelden leert het netwerk patronen en relaties herkennen in de data. Neurale netwerken zijn bijzonder krachtig in het modelleren van complexe, niet-lineaire systemen, en worden daarom veel gebruikt in toepassingen zoals beeldherkenning, spraakherkenning en regeltechniek.



Figuur 1: Illustratie van een neuraal netwerk

2.3 vergelijking PID versus NN-regelaars

Zowel PID-regelaars als neurale netwerken kunnen worden ingezet om systemen te regelen, maar hun werking, toepassingsgebied en eigenschappen verschillen sterk.

2.3.1 Verschillen

- **Regelprincipe:** Een PID-regelaar is gebaseerd op een vaste wiskundige formule die werkt met de fout tussen gewenste en gemeten waarden. Een neuraal netwerk leert juist op basis van voorbeelddata een model van het systeemgedrag.
- **Aanpasbaarheid:** PID-regelaars hebben vaste parameters K_p, K_i, K_d die handmatig of via tuning bepaald worden. Neurale netwerken leren automatisch complexe relaties tijdens het trainen, zonder expliciet geprogrammeerde regels.
- **Complexiteit:** PID-regelaars zijn relatief eenvoudig te implementeren en begrijpen. Neurale netwerken zijn complexer en vereisen meer rekenkracht en data.
- **Flexibiliteit:** PID is minder geschikt voor sterk niet-lineaire systemen of systemen met veel vertraging. Neurale netwerken kunnen deze situaties beter aan, mits voldoende training.
- **Transparantie:** PID-regelaars zijn goed uitlegbaar. Neurale netwerken zijn vaak 'black boxes', waarbij het moeilijk is om te achterhalen waarom een bepaalde beslissing wordt genomen.

2.3.2 Overeenkomsten

- Beide regelmethoden kunnen gebruikt worden voor het aansturen van dynamische systemen.
- Beide maken gebruik van foutsignalen (in neurale netwerken impliciet tijdens het trainen).
- Beide kunnen met feedback werken om de systeemoutput te corrigeren.
- Beide vereisen afstemming of training om goed te presteren in een specifieke toepassing.

2.4 keuze van netwerkarchitecturen

Voordat er in gegaan wordt op de NN-architecturen voor deze toepassing is het belangrijk om met een aantal factoren rekening mee te houden. Deze factoren zijn van belang bij het kiezen van een geschikte NN-architectuur voor het vervangen van een PID-regelaar in een embedded systeem. De factoren die onderandere worden besproken zijn:

- **Real-time vereisten:** Als het systeem real-time reacties vereist, moet de gekozen architectuur snel genoeg zijn om aan deze eisen te voldoen. Dit kan de keuze beperken tot lichtere netwerken of optimalisaties zoals quantisatie.
- **Ondersteuning voor embedded systemen:** De gekozen architectuur moet compatibel zijn met de hardware van het embedded systeem, zoals microcontrollers of FPGA's. Dit kan beperkingen opleggen aan de complexiteit en het type netwerk.
- **Ondersteuning voor machine learning frameworks:** Het is belangrijk om te kiezen voor machine learning frameworks die ondersteuning bieden voor embedded systemen, zoals TensorFlow Lite, Keras of PyTorch Mobile. Deze frameworks zijn ontworpen om neurale netwerken te optimaliseren en te implementeren op beperkte hardware.
- **Trainings- en implementatietijd:** De tijd die nodig is om het neuraal netwerk te trainen en te implementeren, kan ook een factor zijn bij de keuze van de architectuur. Sommige netwerken vereisen meer tijd voor training en afstemming dan andere.

Aan de hand van deze factoren kunnen we een weloverwogen keuze maken voor de netwerkarchitectuur die het beste past bij de specifieke toepassing en de vereisten van het systeem. Het is belangrijk om de juiste balans te vinden tussen complexiteit, prestaties en implementatiegemak, zodat het neuraal netwerk effectief kan functioneren als vervanging voor de PID-regelaar. Hieronder worden enkele architecturen die overwogen kunnen worden voor het vervangen van een PID-regelaar door een neuraal netwerk met hun voor- en nadelen:

- **Recurrente netwerken[3] (RNN):** Deze netwerken zijn ontworpen om tijdsafhankelijke data te verwerken door informatie van eerdere tijdstappen te onthouden. Ze zijn nuttig voor systemen waarbij de huidige output afhankelijk is van eerdere inputs, zoals bij dynamische systemen. RNN's kunnen echter complexer zijn om te trainen en vereisen meer rekenkracht.
- **Convolutioneel Neuraal Netwerk[4] (1D CNN):** Een 1D CNN is specifiek geschikt voor sequentiële of tijdsgebaseerde data, zoals signalen (bijvoorbeeld audio of ECG), tijdreeksen (zoals beursdata of temperatuurmetingen) en reeksen met meerdere features (zoals sensorwaarden). Hoewel convolutionele netwerken vooral bekend zijn uit de beeldverwerking, kunnen 1D CNN's effectief patronen en lokale afhankelijkheden in tijdreeksen herkennen, waardoor ze ook toepasbaar zijn voor regeltoepassingen waarbij de input bestaat uit opeenvolgende meetwaarden.
- **Long Short-Term Memory (LSTM):** Dit is een type RNN dat speciaal is ontworpen om lange-termijn afhankelijkheden te onthouden. LSTM's zijn zeer geschikt voor systemen met complexe dynamiek en tijdsafhankelijke gedragspatronen, maar ze zijn ook complexer en vereisen meer data om goed te presteren.

Voor de toepassing van het vervnagen van een PID-regelaar zijn er aantal kenmerken die belangrijk zijn bij het kiezen van een geschikte netwerkarchitectuur:

- **Eenvoudige structuur:** Voor een microcontroller is het belangrijk dat de netwerkarchitectuur eenvoudig genoeg is om efficiënt te trainen en te implementeren. Complexe netwerken zoals LSTM's kunnen te veel rekenkracht vereisen en zijn moeilijker te optimaliseren voor embedded systemen.
- **Compact model:** Het model moet klein genoeg zijn om op de microcontroller te passen, wat betekent dat het aantal parameters beperkt moet zijn. Dit kan worden bereikt door het aantal lagen en neuronen te beperken, of door technieken zoals quantisatie toe te passen.
- **Snelle inferentie:** De gekozen architectuur moet snel genoeg zijn om real-time voorspellingen te doen, wat betekent dat de inferentietijd laag moet zijn. Dit kan worden bereikt door gebruik te maken van efficiënte activatiefuncties en optimalisaties in de netwerkstructuur.
- **Robuustheid:** Het netwerk moet robuust zijn tegen verstoringen en veranderingen in de systeemparameters. Dit kan worden bereikt door het netwerk te trainen op een breed scala aan scenario's en variaties in de data.
- **Ondersteuning voor embedded systemen:** De gekozen architectuur moet compatibel zijn met de hardware van het embedded systeem, zoals microcontrollers of FPGA's. Dit kan beperkingen opleggen aan de complexiteit en het type netwerk.
- **Ondersteuning voor machine learning frameworks:** Het is belangrijk om te kiezen voor machine learning frameworks die ondersteuning bieden voor embedded systemen, zoals TensorFlow Lite, Keras of PyTorch Mobile. Deze frameworks zijn ontworpen om neurale netwerken te optimaliseren en te implementeren op beperkte hardware.
- **Flexibiliteit in tijdsvenster:** De architectuur moet in staat zijn om te werken met verschillende tijdsvensters van inputdata, zodat het netwerk kan worden aangepast aan de specifieke toepassing en de vereisten van het systeem.

Deze kenmerken helpen bij het selecteren van een netwerkarchitectuur die geschikt is voor het vervangen van een PID-regelaar in een embedded systeem. Het is belangrijk om de juiste balans te vinden tussen complexiteit, prestaties en implementatiegemak, zodat het neuraal netwerk effectief kan functioneren als vervanging voor de PID-regelaar.

uiteindelijk is er gekozen voor een Convolutioneel Neuraal Netwerk (1D CNN) als de meest geschikte architectuur voor het vervangen van een PID-regelaar in een embedded systeem. Deze keuze is gebaseerd op de volgende overwegingen:

- **Eenvoudige structuur:** Een 1D CNN heeft een overzichtelijke en relatief eenvoudige architectuur, waardoor het netwerk efficiënt te trainen en te implementeren is, zeker in vergelijking met complexere netwerken zoals LSTM's.
- **Compact model:** 1D CNN's kunnen worden geoptimaliseerd tot compacte modellen met een beperkt aantal parameters, wat ze geschikt maakt voor embedded systemen met beperkte rekenkracht en geheugen.
- **Snelle inferentie:** Door hun efficiënte structuur bieden 1D CNN's doorgaans snellere inferentie dan recurrente netwerken, wat essentieel is voor real-time toepassingen.
- **Robuustheid en generalisatie:** 1D CNN's zijn effectief in het herkennen van patronen en lokale afhankelijkheden in sequentiële data, waardoor ze goed kunnen generaliseren naar verschillende scenario's en bestand zijn tegen verstoringen.
- **Geschikt voor embedded systemen:** Er zijn diverse machine learning frameworks, zoals TensorFlow Lite en Keras, die 1D

CNN's ondersteunen en optimaliseren voor gebruik op embedded hardware.

- **Brede frameworkondersteuning:** 1D CNN's worden breed ondersteund door populaire machine learning frameworks, wat de ontwikkeling, optimalisatie en implementatie vereenvoudigt.
- **Flexibiliteit in tijdsvenster:** Met 1D CNN's kan eenvoudig worden bepaald over welk tijdsvenster inputdata wordt geanalyseerd, waardoor het netwerk flexibel inzetbaar is voor uiteenlopende regeltoepassingen.

Deze overwegingen maken de 1D CNN een geschikte keuze voor het vervangen van een PID-regelaar in een embedded systeem. Het biedt een goede balans tussen prestaties, eenvoud en implementatiegemak, waardoor het effectief kan functioneren als vervanging voor de PID-regelaar in verschillende toepassingen.

2.5 Vergelijking van prestaties

De prestaties van een neuraal netwerk in vergelijking met een PID-regelaar kunnen op verschillende manieren worden beoordeeld, afhankelijk van de specifieke toepassing en de doelstellingen van de regeling. Enkele belangrijke prestatie-indicatoren zijn:

- **Nauwkeurigheid:** Hoe goed het neuraal netwerk de gewenste output kan voorspellen in vergelijking met de PID-regelaar. Dit kan worden gemeten aan de hand van de fout tussen de voorspelde en werkelijke waarden.
- **Stabiliteit:** Hoe stabiel het systeem is bij het bereiken van de gewenste output. Een PID-regelaar kan soms oscillaties vertonen, terwijl een goed getraind neuraal netwerk deze kan verminderen of elimineren.
- **Snelheid:** Hoe snel het neuraal netwerk kan reageren op veranderingen in de input. Dit is vooral belangrijk in real-time toepassingen waar snelle aanpassingen nodig zijn.
- **Robuustheid:** Hoe goed het neuraal netwerk presteert onder verschillende omstandigheden, zoals verstoringen of veranderingen in de systeemparameters. Een PID-regelaar kan soms gevoelig zijn voor veranderingen in de omgeving, terwijl een neuraal netwerk mogelijk beter kan generaliseren.

De vergelijking van prestaties tussen een neuraal netwerk en een PID-regelaar kan worden uitgevoerd door beide systemen te testen op dezelfde set van inputs en de resultaten te vergelijken. Dit kan bijvoorbeeld worden gedaan door simulaties uit te voeren of door het systeem in de praktijk te testen. Het is belangrijk om de prestaties onder verschillende omstandigheden te evalueren, zoals bij verstoringen of veranderingen in de systeemparameters, om een goed beeld te krijgen van de effectiviteit van het neuraal netwerk in vergelijking met de PID-regelaar.

2.6 Implementatie op embedded systemen

Een van de uitdagingen van het implementeren van een neuraal netwerk is de integratie ervan in embedded systemen. Embedded systemen zijn vaak beperkt in termen van rekenkracht, geheugen en energieverbruik, wat betekent dat de implementatie van een neuraal netwerk op deze systemen specifieke overwegingen vereist. Daarom is het belangrijk om de volgende aspecten in overweging te nemen bij de implementatie van een neuraal netwerk op embedded systemen:

- **Modeloptimalisatie:** Het neuraal netwerk moet worden geoptimaliseerd voor de beperkte rekenkracht en geheugen van het embedded systeem. Dit kan onder meer het verminderen van het aantal parameters, het gebruik van quantisatie of het toepassen van modelcompressie omvatten.
- **Efficiënte inferentie:** De inferentie (voorspellingsfase) van het neuraal netwerk moet efficiënt worden uitgevoerd. Dit kan worden bereikt door gebruik te maken van speciale hardwareversnellers, zoals Tensor Processing Units (TPU's) of Field Programmable Gate Arrays (FPGA's), die zijn ontworpen voor het uitvoeren van neurale netwerken.
- **Real-time prestaties:** Het neuraal netwerk moet in staat zijn om real-time voorspellingen te doen, wat betekent dat de inferentietijd snel genoeg moet zijn om te voldoen aan de vereisten van de toepassing. Dit kan worden bereikt door het netwerk te optimaliseren voor snelheid en door gebruik te maken van efficiënte algoritmen.
- **Compatibiliteit met hardware:** Het neuraal netwerk moet compatibel zijn met de specifieke hardware van het embedded systeem. Dit kan onder meer het gebruik van specifieke bibliotheken of frameworks omvatten die zijn geoptimaliseerd voor de gekozen hardware.
- **Ondersteuning voor machine learning frameworks:** Het is belangrijk om te kiezen voor machine learning frameworks die ondersteuning bieden voor embedded systemen, zoals TensorFlow Lite, Keras of PyTorch Mobile. Deze frameworks zijn ontworpen om neurale netwerken te optimaliseren en te implementeren op beperkte hardware.

Deze microcontroller is geschikt voor embedded toepassingen en biedt voldoende mogelijkheden voor het uitvoeren van machine learning taken. Hieronder volgt een uitgebreid overzicht van de belangrijkste kenmerken van de NXP FRDM-MCXN947[5] Development board, gebaseerd op de officiële NXP-pagina:

- **Processor:** De FRDM-MCXN947[5] is uitgerust met een dual-core ARM Cortex-M33 processor, draaiend op maximaal 150 MHz per core. Deze cores zijn ontworpen voor deterministische prestaties en ondersteunen TrustZone-technologie voor beveiligde toepassingen. De gecombineerde rekenkracht (750 DMIPS per core) maakt deze geschikt voor besturingslogica, preprocessing en embedded AI-taken.
- **Neural Processing Unit (NPU):** De geïntegreerde eIQ Neutron NPU biedt tot 16 GOPS (Giga Operations Per Second) rekenkracht voor machine learning-inferentie. De NPU versnelt modellen met tientallen keren vergeleken met CPU-only uitvoering. Bijvoorbeeld: een keyword spotting-model van 30 kB haalt inferentietijden van enkele milliseconden met een nauwkeurigheid van 97.06%.
- **Geheugen:** De MCU bevat 2 MB on-chip flash in dual-bank configuratie, geschikt voor firmware updates tijdens runtime. Daarnaast is er 512 kB SRAM beschikbaar. Voor grotere modellen of datasets kan via SPI een externe micro-SD-kaart worden toegevoegd.
- **Machine learning frameworks:** TensorFlow Lite for Microcontrollers en het NXP eIQ framework worden ondersteund. Deze frameworks maken optimalisatie via quantisatie, pruning en compressie mogelijk, essentieel voor inferentie op embedded hardware.
- **Real-time prestaties:** Dankzij een systeemtimer, nested vectored interrupt controller (NVIC), DMA-ondersteuning en hardwarematige versnelling van AI-taken kan het systeem real-time inferenties uitvoeren (bijvoorbeeld 1–5 ms per inferentie bij kleine modellen).

Deze kenmerken maken de NXP FRDM-MCXN947 Development board een geschikte keuze voor het implementeren van een neuraal netwerk als vervanging voor een PID-regelaar in embedded systemen. Door rekening te houden met de mogelijkheden en beperkingen

van deze microcontroller, kan het neuraal netwerk worden geoptimaliseerd voor de specifieke toepassing en de vereisten van het systeem.

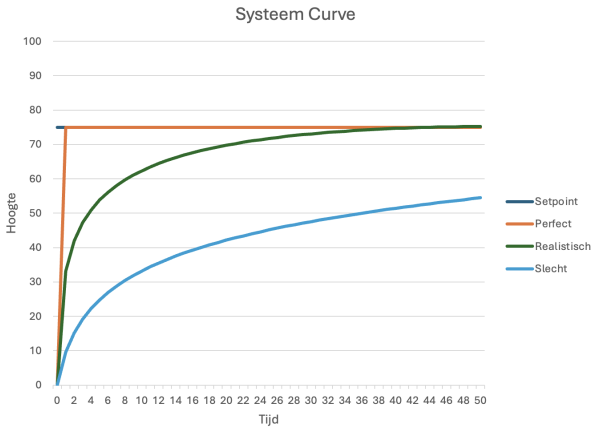
3 Methode

In dit hoofdstuk wordt beschreven op welke wijze het onderzoek is uitgevoerd. De nadruk ligt op de gebruikte software, de strategie voor dataverzameling, de opbouw en training van het neuraal netwerk, en de testomgeving waarin de prestaties van het model zijn geëvalueerd. Er is specifiek rekening gehouden met de beperkingen van de NXP FRDM-MCXN947 microcontroller.

3.1 Manieren van dataverzameling

Het verzamelen van trainingsdata vormt een cruciale stap in het opzetten van een neuraal netwerk voor regeltoepassingen. De gekozen methode heeft directe invloed op de prestaties van het model. Binnen dit onderzoek zijn drie methodes onderzocht voor het trainen van de AI:

1. **PID-kopie:** Het neuraal netwerk wordt getraind op data afkomstig van een werkende PID-regelaar. Het doel is om het gedrag van de PID exact na te bootsen. Deze aanpak vereist een goed afgestemde PID als referentie en resulteert in een AI-model dat functioneert binnen dezelfde prestatiegrenzen als de oorspronkelijke regelaar. Een nadeel is de benodigde werkende PID.
2. **Ideale responscurve:** Hierbij wordt de AI getraind op een theoretisch bepaalde ideale curve, waarin de hoogte van de pingpongbal over de tijd een gewenst verloop volgt (zie bijvoorbeeld zie Figuur 2). Dit biedt meer flexibiliteit bij het verbeteren van de prestaties, maar vereist een nauwkeurige definitie van het gewenste gedrag. Een nadeel is dat kleine afwijkingen in setpoints relatief trage correcties veroorzaken in vergelijking met grote setpointveranderingen.
3. **Iteratieve benadering:** Deze methode combineert de bovenstaande methoden. In de eerste fase wordt het netwerk getraind op data van een PID. Vervolgens worden verbeteringen aangebracht met behulp van de ideale curve. Deze methode maakt snelle en gecontroleerde iteratieve verbeteringen mogelijk.



Figuur 2: manieren van dataverzameling

De initiële dataset werd gegenereerd door middel van simulaties, waarin een bestaande PID-regelaar werd nagebootst. Dit leverde gestructureerde gegevens op, waaronder: tijdstap, setpoint, werkelijke hoogte, fout (setpoint - gemeten waarde), geïntegreerde fout, afgeleide fout en de bijbehorende PWM-waarde. Het uiteindelijke doel is om het netwerk ook te trainen met data uit een fysieke opstelling. Simulatiedata zijn reproduceerbaar en snel te genereren, maar missen systeemeigenschappen die voortkomen uit hardware-imperfecties. Door aanvullend gebruik te maken van data uit de echte opstelling kunnen robuustere modellen worden ontwikkeld. Een bruikbare dataset moet representatief zijn voor diverse systeemcondities. Bij dataverzameling uit de reële opstelling is daarom gelet op variatie in setpoints, verstoringen en afwijkingen in hardwarecomponenten. Het gebruik van meerdere hardware opstellingen elimineert ook de verschillende afwijkingen in hardware en zou dus toepasbaar moeten zijn op verschillende opstellingen. Parameters die consequent zijn meegenomen in de dataset zijn: het setpoint, de actuele hoogte van de bal, de fout, de geïntegreerde fout en de afgeleide fout.

3.2 Software en tools

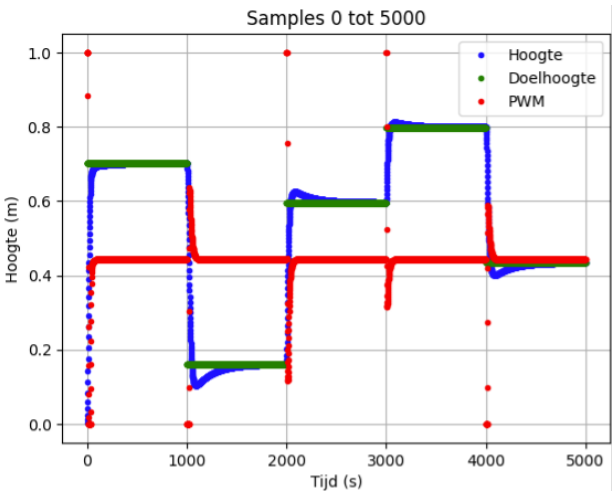
Voor zowel de simulatie van de pingpongbal-opstelling als het opzetten en trainen van het neuraal netwerk is gebruik gemaakt van Python. De volgende libraries en tools zijn ingezet:

- **NumPy** en **Pandas**: voor het verwerken en analyseren van data;
- **Matplotlib** voor visualisatie van resultaten;
- **TensorFlow**[6] en **Keras**: voor het bouwen, trainen en valideren van het neuraal netwerk;
- **Scikit-learn**: voor data preprocessing en evaluatiemethoden;
- **Jupyter Notebook**: voor het ontwikkelen en testen van de code in iteraties.

3.3 Dataverzameling

De trainingsdata voor het neuraal netwerk is verzameld door een simulatie van de bestaande PID-regelaar. In deze simulatie werd het gedrag van de PID-regelaar gelogd terwijl deze een pingpongbal op verschillende hoogtes stabiel probeerde te houden. Hierbij zijn gegevens zoals tijd, doelhoogte, werkelijke hoogte, fout, fout integratie, fout afgeleide en bijbehorende PWM-waarden opgeslagen (zie Figuur 3).

De data werd vervolgens gesplitst in een trainingsset (80%) en een validatieset (20%). De trainingsset werd gebruikt om het model te trainen, terwijl de validatieset diende om de prestaties van het model te evalueren tijdens de training. Dit helpt overfitting te voorkomen en zorgt ervoor dat het model generaliseerbaar blijft naar nieuwe data.



Figuur 3: PID gesimuleerde data

3.4 Neuraal netwerk opzetting

Op basis van de kenmerken van het systeem en de vereisten voor embedded implementatie is gekozen voor een 1D Convolutioneel Neuraal Netwerk (1D CNN). Dit type netwerk is efficiënt in het herkennen van patronen in sequentiële data en heeft relatief lage rekenvereisten.

De architectuur van het netwerk bestaat uit:

- Een inputlaag met vensters van opeenvolgende foutwaarden;
- Een of meerdere 1D convolutionele lagen met ReLU-activatie;
- Een flatten-laag gevolgd door één dense laag;
- Een outputlaag die de benodigde PWM-waarde voorspelt.

De volledige code van het neuraal netwerk is openbaar beschikbaar op GitHub.* De code is geschreven in Python en maakt gebruik van TensorFlow en Keras voor de implementatie van het neuraal netwerk. Daarnaast is er een handleiding toegevoegd, zodat gebruikers eenvoudig stap voor stap door het proces kunnen lopen en het netwerk zelf kunnen opzetten en trainen.

3.5 Trainingsstrategie en hyperparameters tuning

Het model is getraind met behulp van backpropagation en de Adam optimizer. De volgende hyperparameters zijn getest en afgestemd:

- Learning rate: 0.001, 0.0005, 0.0001;
- Batch size: 32 en 64;
- Aantal epochs: 10–100;
- Rechthoekige activatievensters: lengte 5–20 samples.

De performance van het model werd tijdens de training geëvalueerd met behulp van de mean squared error (MSE). Early stopping werd toegepast om overfitting te voorkomen.

3.6 Testopstelling of Simulatieomgeving

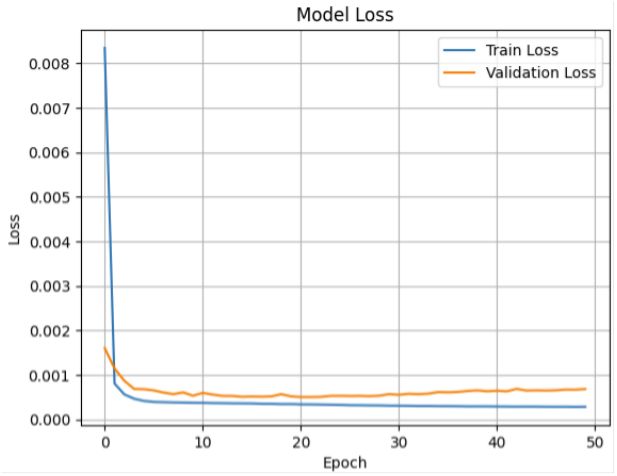
De opstelling is volledig gesimuleerd in Python, waarbij de dynamiek van de pingpongbal in een verticale buis werd gemodelleerd. Hierbij werd rekening gehouden met de krachten op de bal (zwaartekracht, luchtdruk door de ventilator) en de vertraging in het systeem. Zowel de PID-regelaar als het neuraal netwerk werd getest in dezelfde simulatieomgeving zodat een eerlijke vergelijking kon worden gemaakt.

Het gedrag van beide regelstrategieën werd geëvalueerd op responsietijd, stabiliteit, overshoot, en robuustheid bij verstoringen.

4 Resultaten

4.1 Resultaten van de getrainde modellen

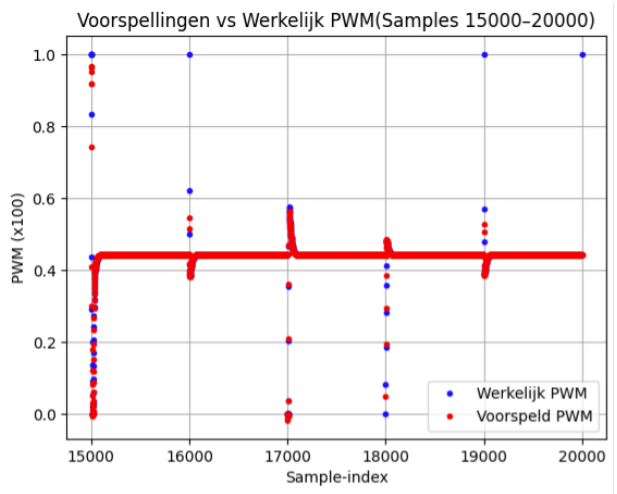
Tijdens het trainen van het neuraal netwerk is gebruik gemaakt van zowel een trainingsset (80%) van de data als een validatieset (20%) om het leerproces van het model te volgen. De prestaties van het model werden geëvalueerd aan de hand van de training en validation loss. De training loss geeft aan hoe goed het model presteert op de trainingsdata, terwijl de validation loss inzicht geeft in de prestaties op nieuwe data, en daarmee de precisie van het netwerk. In figuur 4 is te zien dat het model verschuift naar een training loss van 0.00413 en een validation loss van 0.00160. Deze relatief lage waardes wijzen op een goed getraind model. Het verschil tussen beide verliezen suggereert dat er geen sprake is van significante overfitting of underfitting.



Figuur 4: Training en validation loss van het neuraal netwerk

4.2 vergelijking met de PID-regelaar

Om het model te kunnen vergelijken met de traditionele PID-regelaar, zijn beide regelstrategieën getest in dezelfde simulatieomgeving. Hierbij werd gekeken naar de prestaties op responsietijd, stabiliteit, overshoot en robuustheid. In figuur 5 is een overzicht opgenomen van de meetresultaten. Hieruit blijkt dat het getrainde neuraal netwerk een vergelijkbare prestatie levert als de PID-regelaar in termen van hoogtecontrole van de pingpongbal. Het model wist de gewenste hoogte nauwkeurig te benaderen met een vergelijkbare stabiliteit en minder overshoot in sommige scenario's. De resultaten van deze vergelijking zijn tevens weergegeven



Figuur 5: Vergelijking van de prestaties van het neuraal netwerk en de PID-regelaar

4.3 Effect van variatie in data/robuustheid

In de eerste iteratie van het model werd gekozen voor een grotere netwerkarchitectuur met meerdere verborgen lagen en neuronen per laag. Na overleg met een expert uit het Datalab is besloten om deze complexiteit te reduceren. De reden hiervoor was niet gelinkt aan de beperkingen van de microcontroller, maar aan de aard van het regelprobleem zelf. Het bleek dat het gedrag van een PID-regelaar relatief eenvoudig te modelleren is, waardoor een kleiner model efficiënter bertere prestaties leverde. Bovendien werd vastgesteld dat een kleinere architectuur minder gevoelig is voor overfitting en beter presteert onder variërende invoerdata. Dit verhoogt de robuustheid van

*https://github.com/MITCHEL-Development/PEE51---AIRegelsysteem/tree/main/03_NN-Handleiding

het model, wat essentieel is bij implementatie op fysieke hardware.

4.4 Observatie tijdens simulatie

Tijdens de simulaties viel op dat het neurale netwerk in staat was om consistente prestaties te leveren. De stabiliteit en reactietijd bleven binnen aanvaardbare marges. Hoewel deze observaties grotendeels overeenkomen met de bevindingen uit de vorige paragrafen, bevestigen ze de geschiktheid van het model voor real-time regeltoepassingen.

5 Discussie

5.1 Interpetatie van de resultaten

De verkregen resultaten tonen aan dat een neuraal netwerk in staat is om het gedrag van een PID-regelaar met hoge nauwkeurigheid na te bootsen. Daarbij is ook rekening gehouden met de hardware beperkingen van de microcontroller. Bovendien maakt het ontwikkelde VSC-bestand het mogelijk om eenvoudig variaties van het netwerk te implementeren of te testen met andere datasets. Dit biedt veel flexibiliteit voor toekomstige optimalisaties of uitbreidingen van het systeem.

5.2 Beperkingen van het model

De voornaamste beperkingen liggen bij de beschikbare hardware. Zoals besproken in hoofdstuk 2.6 is het geheugen op de microcontroller beperkt, wat het gebruik van grotere of meer geavanceerde modellen, zoals LSTM-netwerken, belemmert. Ook is er geen rekening gehouden met langdurige inzet of prestaties onder extreme omstandigheden, omdat het model nog niet fysiek is getest.

5.3 Lessen en inzichten

Gedurende het project werd duidelijk dat eenvoudige modelontwerp vaak gunstiger is, zeker in embedded toepassingen. Daarnaast is het belang van kwaliteit, divers en goed gestructureerde trainingsdata niet te onderschatten. Een iteratieve aanpak in modelontwikkeling en code generatie blijkt essentieel om gericht te verbeteren.

5.4 Reflectie op de aanpak

De gekozen methode – waarbij gestart werd met een kopie van de PID en vervolgens verfijning plaatsvond via trainingsdata – bleek tot huidige prestaties effectief. Het combineren van simulatie en validatie leverde waardevolle inzichten op, terwijl de samenwerking met experts uit het Datalab zorgde voor praktische richtlijnen. Wel hadden sommige onderdelen van de code of trainingsstrategie mogelijk eerder geoptimaliseerd kunnen worden.

6 Conclusie en Aanbevelingen

6.1 Beantwoording van de hoofdvraag

Is het mogelijk om een PID-regelaar te vervangen door een AI?
Uit de resultaten van dit onderzoek blijkt dat een neuraal netwerk het gedrag van een PID-regelaar succesvol kan nabootsen. De AI-regelaar presteert vergelijkbaar met de traditionele PID in een gesimuleerde omgeving.

6.2 Samenvatting van de belangrijkste bevindingen

- Een **1D Convolutioneel Neuraal Netwerk (1D CNN)** bleek het meest geschikt voor deze toepassing.
- De prestaties van het model kwamen sterk overeen met die van de PID-regelaar.
- Het model is compact genoeg voor implementatie op een embedded systeem.
- De keuze voor een eenvoudige netwerkarchitectuur leidt zowel tot snelheid als robuustheid.
- Validatie met gesimuleerde data toont aan dat de AI in staat is tot stabiele en nauwkeurige regeling.

6.3 Aanbevelingen voor implementatie

Hoewel TensorFlow Lite ondersteuning biedt voor embedded toepassingen, bleek de EIQ-software van NXP primair gericht op beeldherkenning. Voor praktische implementatie is het daarom aanbevolen om de meegeleverde voorbeeldcode van de NXP FRDM-MCXN947 als basis te gebruiken en deze iteratief, met bijvoorbeeld trunk based development, uit te breiden met het getrainde model. Zo blijft het systeem controleer- en schaalbaar.

6.4 Suggesties voor vervolgonderzoek

Voor vervolgonderzoek wordt het volgende aanbevolen:

- Het model fysiek implementeren op de microcontroller en de real-time prestaties evalueren.
- Verschillende vormen van trainingsdata vergelijken (reëel vs. simulatie).
- De impact van verstoringen, ruis en sensorgebreken op de robuustheid van het model analyseren.
- Experimenteren met andere netwerkarchitecturen, zoals LSTM's, mits de hardware dit toelaat.

Referenties

[1] Stan Franklin. *Artificial Minds*. MIT Press, Cambridge, MA, 1995. Zie p. 293.

[2] GeeksforGeeks. Recurrent neural networks (rnn) - geeksforgeeks, 2024. Accessed: 2024-06-01.

[3] Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education, Upper Saddle River, NJ, 3rd edition, 2009. Zie p. 23.

[4] Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education, Upper Saddle River, NJ, 3rd edition, 2009. Zie p. 201.

[5] NXP Semiconductors. MCXNx4x Family: 32-bit Arm Cortex-M33 @ 150 MHz (Datasheet) (N94x and N54x). https://www.nxp.com/docs/en/data-sheet/MCXNx4xDS.pdf, 2025. Product Data Sheet, Rev. 7 — 25 January 2025.

[6] TensorFlow. Tensorflow tutorials, 2024. Accessed: 2024-06-14.

[7] Abhishek Vijayvargia. *Machine Learning with Python: Design and Develop Machine Learning and Deep Learning Systems Using Real-World Examples*. BPB Publications, 2018.

[8] Wikipedia contributors. Pid-regelaar — wikipedia, de vrije encyclopedie, 2024. Accessed: 2024-06-01.