Systematischer modellbasierter Entwurf einer Reinforcement Learning-basierten neuronalen adaptiven Geschwindigkeitsregelung

Xiaobo Liu-Henke*, Sven Jacobitz, Or Aviv Yarom, Jannis Fritz

Institut für Mechatronik, Ostfalia Hochschule für angewandte Wissenschaften, Salzdahlumer Str. 46/48, 38302 Wolfenbüttel, Deutschland; *x.liu-henke@ostfalia.de

Abstract. In diesem Beitrag wird der systematische modellbasierte Entwurf einer Reinforcement Learning-basierten neuronalen adaptiven Geschwindigkeitsregelung beschrieben. Ausgehend von einer Einleitung und der Zusammenfassung aktueller Grundlagen, werden Entwurfsmethoden für intelligente Fahrfunktionen vorgestellt. Der Fokus liegt auf der erstmaligen Vorstellung einer neuartigen Entwurfsmethodik für Künstliche Neuronale Netze in der Regelungstechnik. Diese wird anschließend am Beispiel einer adaptiven Geschwindigkeitsregelung vollständig angewendet und validiert.

Einleitung

Der vom Europäischen Fonds für regionale Entwicklung (EFRE) geförderte Innovationsverbund *autoMoVe* (*Dynamisch konfigurierbare Fahrzeugkonzepte für den nutzungsspezifischen autonomen Fahrbetrieb*), hat die Entwicklung eines autonomen, modularen und elektrischen Fahrzeugkonzeptes zum Ziel. Durch den Austausch anwendungsspezifischer Module zur Laufzeit soll eine Vielzahl von Anwendungen vom innerbetrieblichen Gütertransport bis zur Personenbeförderung im Straßenverkehr autonom realisiert werden. Im Rahmen dieses Forschungsvorhabens fokussiert das Teilprojekt der Ostfalia *autoEVM* (*Ganzheitliches elektronisches Fahrzeugmanagement für autonome Elektrofahrzeuge*) die modellbasierte Entwicklung innovativer intelligenter Algorithmen und Funktionen für den autonomen Fahrbetrieb.

Mit einer höheren Automatisierung des Fahrbetriebs geht auch ein Anstieg der Anforderungen an das Fahrzeug bzw. die automatisierten Fahrfunktionen einher. Aktuelle Funktionen und Algorithmen, die auf Methoden der Regelungstheorie oder der klassischen Informationsverarbeitung basieren, können diesen nicht mehr in vollem Umfang gerecht werden [1]. Deshalb stellt Künstliche Intelligenz (KI) in diesem Vorhaben bzw. für viele Domänen, die an der Entwicklung und Nutzung intelligenter, automatisierter Fahrzeuge beteiligt sind eine Schlüsseltechnologie dar [2].

Unabhängig von der Art der Informationsverarbeitung sind moderne Fahrzeuge und Fahrfunktionen komplexe mechatronische Systeme mit einem hohen internen und externen Vernetzungsgrad. Zur Beherrschung dieser Komplexität im Entwicklungs- und Absicherungsprozess ist ein systematisches Entwurfsvorgehen unabdingbar. Dabei wird zum einen auf eine in der Mechatronikforschung etablierte verifikationsorientierte und simulationsgestützte Entwurfsmethodik zurückgegriffen. [3] KI-Algorithmen, insbesondere die hier eingesetzten Künstliche Neuronale Netze (KNN), unterscheiden sich in ihrer Funktionsweise und ihrem Auslegungsprozess stark von regelungstheoretischen Ansätzen. Daher wird in dieser Veröffentlichung zum anderen eine neuartige Entwurfsmethodik für KNN angewandt und erstmalig vorgestellt.

Als Teilziel des oben genannten Forschungsvorhabens wird in diesem Beitrag die erwähnte Enwurfsmethodik für KNN auf ein System zur automatisierten Längsführung im Sinne einer intelligenten adaptiven Geschwindigkeitsregelung angewendet. Intelligent bezieht sich zum einen auf den Einsatz der, an der Funktionsweise des menschlichen Gehirns angelehnten, KNN und zum anderen auf den selbstlernenden bzw. erfahrungsbasierten Trainingsansatz des Reinforcement Learning.

1 Stand der Technik

1.1 Intelligente Fahrfunktionen zur automatisierten Längsführung

Mit steigender Anzahl und Vernetzung von Fahrerassistenzsystemen (FAS) delegiert der menschliche Fahrer sukzessive Fahraufgaben an das Fahrzeug, bis er beim autonomen Fahrbetrieb schließlich zum Passagier wird. Dann spricht man nicht mehr von FAS, sondern von (automatisierten) Fahrfunktionen.

Ein Beispiel für eine solche automatisierte Fahrfunktion ist die hier betrachtete adaptive Geschwindigkeitsregelung (*engl. Adaptive Cruise Control* (ACC)) als Weiterentwicklung der Geschwindigkeitsregelung. Im Gegensatz zur einfachen Geschwindigkeitsregelung, bei der lediglich ein Abgleich zwischen der Vorgabegeschwindigkeit des Fahrers und der Ist-Geschwindigkeit des Fahrzeugs stattfindet, kann die ACC durch Auswertung von Umfeldsensorik die Vorgabegeschwindigkeit nach unten hin überschreiben. Durch anschließende Vorgabe von Sollwerten an unterlagerte Systeme der Längsdynamik (Antrieb, Bremse, Getriebe), hält das Fahrzeug einen geschwindigkeitsabhängigen Sicherheitsabstand zu vorausfahrenden Fahrzeugen und Objekten ein. [4]

Für die Erkennung vorausfahrender Hindernisse sowie die anschließende Bestimmung der relativen Geschwindigkeit und des Abstands kommen typischerweise Radarsensoren zum Einsatz [5]. Moderne Systeme nutzen häufig zusätzlich Kamera- und Lidarsensoren zur Objekterkennung. Ein weiterer Ansatz ist der Einsatz drahtloser Vehicle-to-everything (V2X)-Kommunikation zur Realisierung eines kooperativen Fahrbetriebs [8].

Im regelungstechnischen Sinne ist die ACC ein kaskadiertes System, bei dem die innere Kaskade eine Geschwindigkeit einregelt, die von der äußeren Kaskade vorgegeben wird, um den Sicherheitsabstand einzuregeln. In der Praxis existieren jedoch einige Fahrfunktionen, die zwischen dem Abstandhalten und der reinen Geschwindigkeitsregelung umschalten. Für einen hochautomatisierten Fahrbetrieb sind höherwertige und ganzheitliche Funktionen jedoch von Vorteil. [5]

In der aktuellen Literatur sind viele Ansätze, wie eine klassische PI- [6], eine nichtlineare modellprädiktive [7] oder auch eine neuro-fuzzy Regelung [4] für eine ACC mit jeweils spezifischen Vor- und Nachteilen zu finden. In diesem Beitrag soll ein neuartiger ganzheitlicher Ansatz für eine ACC mit KNN und Reinforcement Leaning untersucht werden.

1.2 Grundlagen Künstlicher Neuronaler Netze und Maschinellen Lernens

Der Begriff KI umfasst eine Vielzahl unterschiedlichster Methoden und Algorithmen, die sich mit der selbstständigen und automatisierten Lösung von Problemen befassen [2]. KNN und ML bilden ein Teilgebiet der KI, das sich in zahlreichen Problemen verschiedenster Gebiete, auch beim autonomen Fahrbetrieb, als geeignet erwiesen hat. Deshalb fokussiert sich dieser Beitrag auf dieses Teilgebiet. Die zahlreichen positiven Eigenschaften von KNN und ML, wie Anpassungsfähigkeit, Fehlerresistenz, Vielseitigkeit und vor allem Lernfähigkeit sind auf ihre Anlehnung an die Struktur und Funktionsweise des menschlichen Gehirns zurückführen.

Analog zur Biologie sind (künstliche) Neuronen Verarbeitungseinheiten, die Eingangsreize über gewichtete Verbindungen kumulieren und mit Hilfe einer Aktivierungsfunktion eine Ausgabe berechnen. Durch die Zusammenschaltung mehrerer Neuronen in mindestens zwei Schichten entsteht das KNN. Üblich sind Kombinationen von bis zu einigen Hundert Neuronen in bis zu über Einhundert Schichten. Dabei sind nicht nur beliebige vorwärtsgerichtete, sondern auch zeitlich rückgekoppelte Verbindungen im KNN möglich. Die optimale Architektur eines KNN lässt sich bisher nicht analytisch bestimmen [9]. Deshalb ist neben Erfahrung und Versuchsreihen auch eine systematische Entwurfsmethodik notwendig, um eine geeignete Architektur im Zielkonflikt zwischen Rechenaufwand und Leistungsfähigkeit zu finden. Die Anzahl, Verschaltung und Gewichtung der Verbindungen charakterisiert die „Intelligenz“ eines KNN. Allgemein gesagt, bedeuten mehr Neuronen und Verbindungen eine höhere Leistungsfähigkeit des KNN, bei gleichzeitig steigendem Rechenaufwand.

Genau wie ein menschliches Gehirn, muss das KNN eine Aufgabe zunächst lernen bzw. trainieren. Diese Begriffe bezeichnen die Anpassung der Verbindungsgewichte. Im Umfeld des autonomen Fahrens sind dafür das Supervised Learning (SL) und das Reinforcement Learning (RL) relevant. Beim SL werden dem KNN Eingangsdaten und die zugehörige Ausgabe präsentiert. Das KNN lernt iterativ den Zusammenhang zwischen den beiden Größen [10]. Dieses Lernverfahren eignet sich z.B. besonders für bildbasierte Objekterkennung [11]. Beim RL lernt das KNN sukzessive aus der Erfahrung vergangener Sequenzen die optimale Strategie im Sinne einer vorgegeben Belohnungsfunktion [10]. Dieses Verfahren wird eingesetzt, wenn keine Trainingsdaten verfügbar sind, z.B. bei der automatisierten Fahrzeugführung [12]. SL und RL sind Oberkategorien von Lernverfahren, mit diversen Trainingsalgorithmen. Genau wie die KNN Architektur lassen sich die optimalen Trainingsalgorithmen bzw. deren Parameter nicht analytisch bestimmen. So sind auch hier Erfahrung und Versuche erforderlich.

2 Entwurfssystematik

2.1 Modellbasierte Reglerauslegung

Die Komplexität moderner Fahrzeuge steigt durch den höheren internen und externen Vernetzungsgrad sowie die wachsende Anzahl intelligenter und leistungsfähiger Hard- und Softwarekomponenten stetig an. Zur Beherrschung der Systemkomplexität und frühzeitigen Fehlervermeidung bei der Auslegung der Informationsverarbeitung, ist eine ganzheitliche Entwurfsmethodik unverzichtbar. Daher hat sich die durchgängige, verifikationsorientierte, modellbasierte Entwurfsmethodik, basierend auf dem Rapid Control Prototyping (RCP) und Model-in-the-Loop-(MiL), Software-in-the-Loop-(SiL) und Hardware-in-the-Loop-(HiL) Simulationen etabliert. [13]

Die Methodik baut auf funktionsorientierten physikalischen Modellen einer Regelstrecke auf. Die Regelungsfunktion wird anschließend in Abhängigkeit des Systemverhaltens simulativ ausgelegt und frühzeitig in MiL-Simulationen validiert. Um manuelle Programmierung zu umgehen, werden Modell und Regelungsfunktion in blockschaltbildbasierten Programmiersprachen entwickelt. Der anschließend automatisch generierte Funktionscode wird in SiL-Simulationen erneut gegen das Streckenmodell getestet. HiL-Simulationen dienen zur weiteren Validierung und Optimierung der Informationsverarbeitung mit echtzeitfähigen Simulationsmodellen und realen Teilkomponenten des zu regelnden Systems.

Der verifikationsorientierte und iterative Ansatz dieser Methodik unterstützt den Entwicklungsprozess auch in der herausfordernden Aufgabe der Absicherung. Die Methodik adressiert die Schwächen der klassischen auf physischen Prototypen basierenden Validierung, wie einem hohem Ressourcenaufwand oder Sicherheitsrisiken für Mensch, Maschine und Umwelt. Durch ihren virtuellen Charakter sparen MiL-, SiL- und HiL-Simulationen Zeit und Kosten [14]. Sie ermöglichen jederzeit durchführbare und reproduzierbare Tests ohne direkte Abhängigkeit von physischen Prototypen, Tageszeiten oder menschlichen Experten. So lassen sich Simulationsdurchläufe, für verschiedene Funktionsvarianten oder Szenarien automatisiert durchführen. Damit eignet sich diese Methodik auch besonders für das Training von KNN. Denn das sogenannte maschinelle Lernen verläuft, bis auf seltene Ausnahmen, immer iterativ.

Virtuelle Entwurfsmethoden wie diese bilden die Grundlage für viele intelligente Systeme, wie hochautomatisierte Fahrzeuge. Mit prototypenbasierten Tests sind die benötigten Hunderttausenden Testkilometer nicht mit vertretbarem Zeit- und Kostenaufwand zu leisten. [14]

2.2 Systematische Auslegung von KNN

Die im vorherigen Abschnitt 2.1 vorgestellte Entwurfssystematik zum modellbasierten Reglerentwurf ist eine übergeordnete Methodik. Die Auslegung eines KNN kann analog zu einer Reglerauslegung mit dynamischer Kompensation als ein Teil dieser Methodik betrachtet werden. KNN Auslegung ist ein komplexer Prozess, der aus vielen Entwurfsentscheidungen besteht, welche einzeln oder in Kombination einen großen Einfluss auf das Entwicklungsergebnis haben. Diese Entscheidungen sind über den gesamten Entstehungsprozess der KNN verteilt und betreffen z.B. die Systemstruktur (End-2-End-Funktion / Kombination von Sub-Funktionen), die Lernart (SL / RL), die Netzarchitektur, die Parametrierung des Trainingsalgorithmus oder auch die Bewertung des KNN.

Die Besonderheit hierbei liegt darin, dass im Gegensatz zu den sehr systematisch-analytischen Auslegungsverfahren der klassischen und modernen Regelungstechnik, die Entscheidungen bzw. deren Auswirkungen bei der KNN Auslegung für den menschlichen Verstand meist nicht mathematisch-logisch durchdringbar sind. Infolgedessen existiert bisher kein systematisches Auslegungsverfahren für KNN [9]. Obwohl man in aktueller Literatur viele Anwendungen von KNN finden kann, sind Informationen zu den jeweiligen Auslegungsprozessen rar. Wenn Angaben dazu gemacht werden, verfolgen die Autoren oft einen ergebnisorientierten auf Erfahrungen basierenden empirischen Ansatz mit wenig Systematik.

Das in Abbildung 1 dargestellte Vorgehensmodell stellt einen ersten Lösungsansatz vor, um den Auslegungsprozess von KNN zu systematisieren. Der Ausgangspunkt ist die regelungstechnische Problemspezifikation. Sie unterscheidet sich kaum von der klassischen Regelungstheorie. Beispielsweise wird auch hier - vor allem beim RL - typischerweise modellbasiert gearbeitet. Die Besonderheit liegt in der Wahl der Systemstruktur und Schnittstellen. Während sich diese bei klassischen Reglern oft aufgrund physikalischer Gegebenheiten ergeben, können sie bei KNN theoretisch frei gewählt werden. Anschließend muss eine für das jeweilige Problem geeignete Leranrt bzw. ein konkreter Lernalgorithmus gewählt werden. Es existieren viele Lernalgorithmen, mit spezifischen Vor- und Nachteilen, die für jede Anwendung neu ausgewählt werden müssen. Beim RL unterscheidet man grundsätzlich zwischen gradienten-basierten und gradientenfreien Algorithmen.

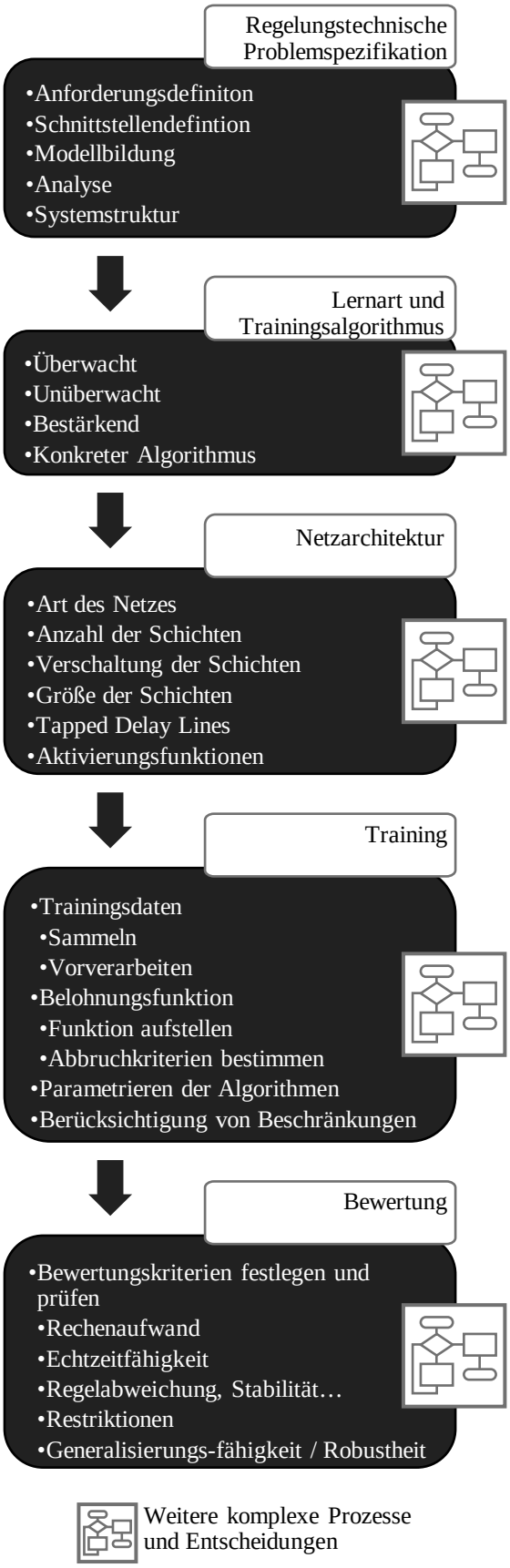

Abbildung 1: Vorgehensmodell zur systematischen Auslegung von KNN

Die Auswahl der Netzarchitektur ist der aufwendigste Teil der KNN Auslegung und hat sich zu einem eigenen modernen Forschungsgebiet entwickelt. Dabei versucht man die Architekturauswahl als Suchproblem zu formulieren und durch ein überlagertes Optimierungsverfahren automatisiert zu lösen. Man spricht von der Neural Architecture Search (NAS). [15] Nachdem die Netzarchitektur feststeht, kann mit dem Training, dem eigentlich Kern der Auslegung, fortgefahren werden. Dabei wird das Verhalten des KNN hinsichtlich der ursprünglichen Aufgabe optimiert. In diesem Schritt müssen vor allem die Hyperparameter der Trainingsalgorithmen eingestellt werden. Der letzte Schritt ist die Bewertung des KNN hinsichtlich der ursprünglichen Anforderungen und unter Berücksichtigung seines Einsatzzwecks, z.B. auf einem Echtzeitsystem.

Wie bereits bei der NAS angedeutet, bildet jeder Block im Vorgehensmodell aus Abbildung 1 einen übergeordneten Prozessschritt und beinhaltet jeweils weitere komplexe Prozesse und Entscheidungen. Da sich dieser Beitrag auf die Anwendung eines KNN als ACC fokussiert, werden nur einige davon später in diesem Beitrag konkreter aufgegriffen.

3 Systematischer modellbasierter Entwurf der neuronalen adaptiven Geschwindigkeitsregelung

Ziel des Beitrags ist der systematische modellbasierte Entwurf einer Reinforcement Learning-basierten neuronalen adaptiven Geschwindigkeitsregelung. Dieser Prozess wird nun nach der vorgestellten Entwurfsmethodik aus Abbildung 1 vorgestellt.

3.1 Regelungstechnische Problemspezifikation

Anforderungsdefinition

Die hier vorgestellte und entwickelte Funktion zur automatisierten Längsführung soll die folgenden Anforderungen erfüllen:

1. Eine vom Fahrer vorgegebene Geschwindigkeit einhalten und keinesfalls überschreiten
2. Einen 2-Sekunden-Sicherheitsabstand zu einem möglicherweise vorausfahrenden Fahrzeug einhalten
3. Den Sicherheitsabstand nicht bzw. nur wenig und kurz unterschreiten

4. Beliebigen Geschwindigkeitsprofilen folgen, auch beim Einhalten des Sicherheitsabstands
5. Geschwindigkeit und Abstand möglichst ohne Schwingungen einhalten
6. Das Ego Fahrzeug soll Beschleunigungen im Bereich $-3\,\frac{m}{s^2}$ bis $2\,\frac{m}{s^2}$ realisieren
7. Auf KNN und RL basieren

Systemstruktur und Schnittstellendefinition

Abbildung 2 zeigt die Systemstruktur, der hier entworfenen neuronalen adaptiven Geschwindigkeitsregelung inklusive Systemumgebung schematisch. Ausgangspunkt ist ein Führungsfahrzeug (*engl. lead car*), das sich mit einer beliebigen Geschwindigkeit v_{lead} bewegt und damit seine Position x_{lead} verändert. Das eigentliche Zentrum der Struktur ist aber das Ego-Fahrzeugmodell, das basierend auf Beschleunigungs- F_A und Bremskräften F_B die Geschwindigkeit v_{ego} und Position x_{ego} des Ego-Fahrzeugs berechnet. Diese Positionen und Geschwindigkeiten des Ego- und Lead Car, dienen dem Radarsensor mit „eingebautem“ Preprocessing, um innerhalb seiner Reichweite von 150 m die Eingangsgrößen für das KNN zu bestimmen. Diese sind:

- der Geschwindigkeitsfehler v_{err}, also die Differenz von der vom Fahrer vorgegebenen Geschwindigkeit v_{set} und der tatsächlichen Ego-Geschwindigkeit
- das Integral des Geschwindigkeitsfehlers über die Zeit $\int v_{err}$
- die Ego-Geschwindigkeit v_{ego}
- der Abstandsfehler d_{err}, also die Differenz zwischen dem sicheren Abstand d_{safe} und dem tatsächlichen Abstand zwischen den Fahrzeugen d.

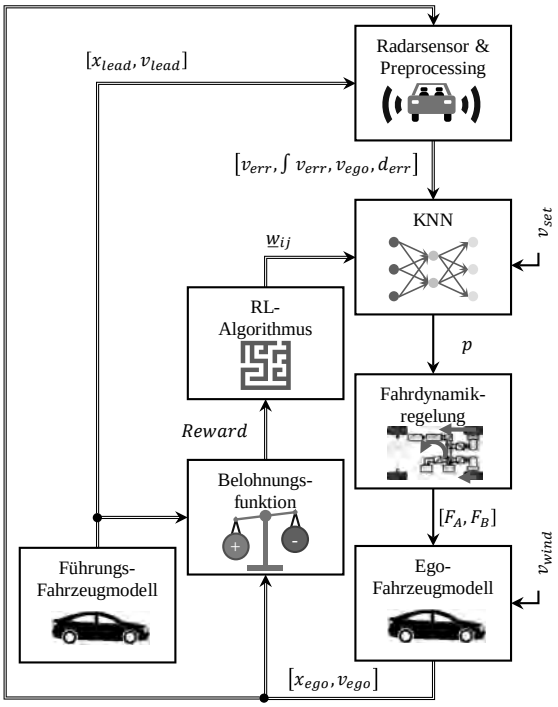

Abbildung 2: Funktionsstruktur der Reinforcement Learning-basierten neuronalen adaptiven Geschwindigkeitsregelung

Das KNN berechnet dann mithilfe dieser Eingangsgrößen eine Vorgabe für die Fahr- und Bremspedalstellung $p \in \{-1 \leq p \leq 1 | \mathbb{R}\}$, welche von einer unterlagerten Fahrdynamikregelung in die oben genannten Längskräfte umgesetzt wird.

Dies sind alle Komponenten, die das System in der Anwendungsphase, also nach dem Training verwendet. Während des Trainings, werden zusätzlich zwei weitere Komponenten, benötigt. Die vom Entwickler vorgegebene bzw. auszuarbeitende Belohnungsfunktion, welche basierend auf Daten der Systemkomponenten einen sogenannten $Reward$ berechnet, um das KNN im Training zu bewerten. Der RL-Algorithmus ist ein Optimierungsverfahren, das die Gewichte $\underline{w}_{i,j}$ des KNN so anpasst, dass der erwartete $Reward$ in der kommenden Episode (ein Simulationsdurchlauf), maximiert wird.

Modellbildung

Das Gesamtsystem aus Abbildung 2 wird in der Simulationsumgebung für die automatisierte Modellkonfiguration zur Auslegung und Absicherung KI-basierter Fahrfunktionen aus [16] modelliert, simuliert und trainiert.

Um das Verhalten der Ego und Lead Fahrzeuge abzubilden, dient ein klassisches Längsdynamikmodell:

$$m \cdot \ddot{x} = F_A - \underbrace{c_w \cdot A \cdot \frac{\rho}{2} \cdot \dot{x}^2}_{Luftwid.} - \underbrace{m \cdot \sin(\alpha)}_{Steigungswid.} - \underbrace{m \cdot \cos(\alpha) \cdot f_R}_{Rollwid.} - F_B. \quad (1)$$

Die Variable für die Fahr- und Bremspedalstellung p wird auf die entsprechenden Kräfte F_A und F_B skaliert. Um einen Stillstand und eine Rückwärtsfahrt simulieren zu können, wurden in der Simulationsumgebung weitere Mechanismen implementiert, auf die in diesem Beitrag nicht eingegangen wird.

Der Radarsensor mit Preprocessing wird aktiviert, sobald die Distanz zwischen den Fahrzeugen die Reichweite des Sensors unterschreitet. Hier werden zum einen situationsbedingte Sollwerte für Abstand und Geschwindigkeit berechnet. Beispielsweise gilt für die Sollgeschwindigkeit des Ego Car $\min(v_{set}, v_{ego})$, wenn sich das Ego Car im Bereich von d_{safe} zum Lead Car oder

darunter befindet. Zum anderen werden die berechneten Sollgrößen auch mit den Ist-Größen abgeglichen und Abweichungen für die Distanz und die Ego Geschwindigkeit berechnet. So liefert dieser Funktionsbestandteil nicht nur die Eingangsgrößen für das KNN, sondern auch Basisgrößen für die Belohnungsfunktion.

3.2 Lernart und Trainingsalgorithmus

Prinzipbedingt kommen für diese Anwendung nur Algorithmen des RL infrage. Um einerseits eine optimale ACC Funktion zu erzeugen und andererseits weitere Erfahrung für den systematischen Entwurf von KNN zu sammeln, sollen hier verschiedene Trainingsalgorithmen verwendet und verglichen werden. In Vorversuchen mit niedrigeren Anforderungen kamen die folgenden Trainingsalgorithmen zum Einsatz:

- Genetischer Algorithmus (GA)
- Particle Swarm Optimization (PSO)
- Deep Deterministic Policy Gradients (DDPG).

Bei den beiden ersten Verfahren handelt es sich um gradientenfreie, evolutionsbasierte Algorithmen. Diese Klasse von Algorithmen entwickelt sich in der Regel stetig in Richtung einer höheren Belohnungsfunktion, konvergiert aber nicht immer in ein echtes Optimum. Ihr Vorteil ist, dass Sie in der Regel recht schnell zu einem guten Ergebnis gelangen. Eine Besonderheit des GA ist, dass dieser recht zufallsbehaftet ist und bei richtiger Parametrierung eher dazu neigt, das globale Optimum zu finden, als andere Algorithmen. Der DDPG hingegen, ist ein gradientenbasierter Algorithmus, der ziemlich sicher in ein lokales Optimum konvergiert. Dafür benötigt er jedoch meist mehr Zeit. Außerdem wird nur selten das globale Optimum erreicht.

Die Versuchsreihe zeigte, dass der DPPG mit hoher Wahrscheinlichkeit die besten Ergebnisse erzielt. Um die Ergebnisse vergleichbar zu machen, wurde für alle Algorithmen die gleiche rudimentäre Belohnungsfunktion verwendet. Während diese Versuchsreihe bereits dafür genutzt wurde die Hyperparameter der Algorithmen zu optimieren, wurde die Belohnungsfunktion an späterer Stelle detailliert entworfen.

3.3 Netzarchitektur

Durch die Wahl des DDPG als Trainingsalgorithmus müssen zwei KNN Architekturen gewählt werden. Denn hier werden ein *actor* und ein *critic* Netzwerk benötigt. Das *actor* Netz ist das KNN, welches die spätere Regelungsaufgabe übernimmt. Das *critic* Netz dient als Übersetzer zwischen der Belohnungsfunktion und dem Algorithmus zur Aktualisierung der Verbindungsgewichte. Es wird nur während der Trainingsphase benötigt.

Das *critic* Netzwerk kann meist als einfaches vorwärtsgerichtetes Netz mit Gleichrichterfunktionen als Aktivierung gewählt werden. Daher wurde hier ein solches KNN mit fünf schichten als *critic* Netzwerk festgelegt. Die Eingangsgrößen sind neben denen des *actor* Netzes auch dessen Ausgang.

Die Architektur des *actor* Netzwerks wurde mittels einer NAS, die auf einem GA basiert automatisiert optimiert. Das Ergebnis ist ein vorwärtsgerichtetes KNN mit vier Schichten. In den ersten drei Schichten befinden sich jeweils 48 verdeckte Neuronen sowie Gleichrichterfunktionen zur Aktivierung. In der letzten Schicht befindet sich nur noch ein Neuron mit einer $tanh$-Aktivierung.

3.4 Training

Nachdem die Netzarchitekturen festgelegt wurden, muss nun die Belohnungsfunktion entworfen werden, die das letztendliche Verhalten des KNN bzw. des Ego Car bestimmt. Diese besteht aus vier Termen:

- $-0{,}1 \cdot (v_{err})^2$: Bestrafung von Geschwindigkeitsfehlern, damit die Sollgeschwindigkeit eingehalten wird
- $-(a)^2$: Bestrafung von Beschleunigungen, damit Schwingungen und abrupte Geschwindigkeitsänderungen vermieden werden
- $-0{,}1 \cdot (d_{err})^2$: Bestrafung von Abstandsfehlern, damit der Sollabstand eingehalten wird
- $+v_{err}\ f\ddot{u}r\ v_{err} \leq 0{,}25\,\frac{m}{s}$: Belohnung von kleinen Geschwindigkeitsfehlern, damit die Sollgeschwindigkeit eingehalten wird.

In einem iterativen Prozess durchläuft das Ego Car nun mehrere Episoden in denen das *actor* Netz immer wieder als Längsdynamikregler eingesetzt wird und für das Verhalten einen entsprechenden Reward erhält. Abbildung 3 zeigt den Verlauf der Rewards über die Episoden sowie den durchschnittlichen Reward. Der Reward ist zu Beginn des Trainings sehr negativ, da das KNN sich ohne Vorwissen noch nicht gut verhält. Der Reward verbessert sich schnell in den Bereich um ca. -600 und steigert sich dann langsam bis das KNN in der besten Episode am Ende ca. -91 Reward-Punkte erreicht. Zwischendurch gibt es vereinzelte Einbrüche, was auf eine falsche Gewichtsveränderung hindeutet, aber vom Algorithmus wieder ausgeglichen werden kann. Dass der beste Reward bei -91 liegt, sagt nicht, dass das Verhalten des KNN schlecht ist. Dies ist lediglich auf das Design der

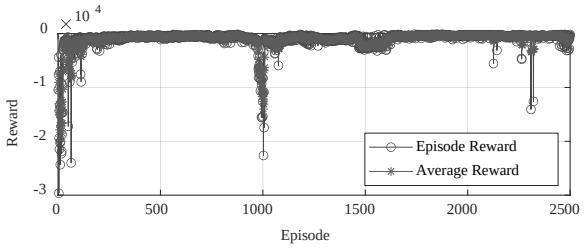

Abbildung 3: Verlauf des Reward beim Training

Belohnungsfunktion zurückzuführen. Der Verlauf der Rewards ist also entscheidender als sein absoluter Wert. Daher ist die Bewertung nach dem Training essentiell.

3.5 Bewertung

Die Bewertung ist der letzte Schritt bei der Auslegung des KNN für die neuronale adaptive Geschwindigkeitsregelung. Ein besonderer Aspekt, der bei der Bewertung berücksichtigt werden muss, ist die sogenannte Generalisierungsfähigkeit. Das bedeutet, dass das KNN das erlernte Wissen auf neue, bisher unbekannte Fragestellungen anwenden soll. Um die Generalisierungsfähigkeit beurteilen zu können, muss sich das Testszenario zur Bewertung also von der Trainingssituation unterscheiden.

Während des Trainings lag v_{set} bei $100\,\frac{km}{h}$ und das Lead Car fuhr mit einem Beschleunigungssinus vorneweg. In der Testsituation wird v_{set} auf $108\,\frac{km}{h}$ erhöht und das Fahrdynamikmodell des Lead Car durch ein „hartes“ Geschwindigkeitsprofil ersetzt, um die Situation für das KNN weiter zu erschweren. Das Geschwindigkeitsprofil des Lead Car ist im oberen Teil der Abbildung 4 in rot zu sehen, v_{set} in blau. Die gelbe Kurve zeigt die Geschwindigkeit des Ego Car. Im unteren Teil der Abbildung sind die Abstände d_{safe} und d über der Zeit aufgetragen.

Zu Beginn wurde das Ego Car bewusst sehr nah ($d < d_{safe}$) hinter dem Lead Car platziert, sodass es zunächst seine Geschwindigkeit halten muss, um den Sicherheitsabstand einzuhalten. Anschließend folgt das Ego Car dem Geschwindigkeitsverlauf des Lead Car sehr gut und stabil, bis das Lead Car bei ca. $24\,s$ auf $120\,\frac{km}{h}$, also oberhalb von v_{set} beschleunigt. Das Ego Car überschreitet, wie gewünscht, v_{set} nicht und der Abstand wird größer. Anschließend bremst das Lead Car abrupt ab und das Ego Car bremst ebenfalls stark, um den Sicherheitsabstand einzuhalten, ohne die Beschleunigungsanforderung (Abschnitt 3.1) zu verletzen. Zum Schluss beschleunigt das Lead Car erneut. Das Ego Car regelt v_{set} stabil ein.

Das bedeutet, dass das KNN die adaptive Geschwindigkeitsregelung generalisierungsfähig erlernt hat und alle Anforderungen zufriedenstellend und erwartungsgemäß erfüllt. Somit kann auch die Entwurfsmethodik als validiert betrachtet werden.

4 Zusammenfassung und Ausblick

In diesem Beitrag wurde der systematische modellbasierte Entwurf einer Reinforcement Learning-basierten neuronalen adaptiven Geschwindigkeitsregelung vorgestellt. Ausgehend von einer Einleitung wurde der Stand der Technik und die zugrundeliegende Entwurfsmethodik zusammengefasst. Diese bezieht sich auf die modellbasierte Reglerauslegung und die systematische Auslegung von KNN. Letztere wurde in diesem Beitrag erstmalig als Neuheit vorgestellt.

Anschließend erfolgte die Anwendung und Validierung dieser Methodik am Beispiel einer neuronalen adaptiven Geschwindigkeitsregelung. Der Auslegungsprozess wurde detailliert beschrieben und die entstandene Funktion intensiv ausgewertet und anhand der Anforderungen validiert.

Zukünftige Arbeitsschritte umfassen die weitere Validierung der Entwurfsmethodik sowie die Auslegung weiterer automatisierter Fahrfunktionen.

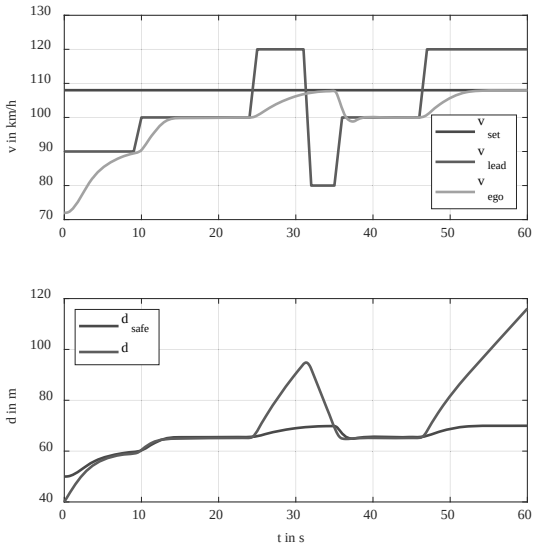

Abbildung 4: Simulationsergebnis der neuronalen adaptiven Geschwindigkeitsregelung

Danksagung

Diese Veröffentlichung entstand aus dem Teilprojekt "autoEVM" (*Ganzheitliches elektronisches Fahrzeugmanagement für autonome Elektrofahrzeuge*) (ZW 6-85030889), im Rahmen des Innovationsverbundes "autoMoVe" (*Dynamisch konfigurierbare Fahrzeugkonzepte für den nutzungsspezifischen autonomen Fahrbetrieb*), welcher vom Europäischen Fonds für regionale Entwicklung (EFRE) gefördert und vom Projektträger NBank verwaltet wird .

EUROPÄISCHE UNION

EUROPA FÜR NIEDERSACHSEN
Europäische Förderung für die niedersächsischen Regionen

Literatur

[1] Milz S., Schrepfer J. Is artificial intelligence the solution to all our problems? Exploring the applications of AI for automated driving. In: Bertram T. (eds) *Automatisiertes Fahren 2019*. Springer Vieweg, Wiesbaden, 2020.

[2] Schiekofer, P. et al. Maschinelles Lernen für das automatisierte Fahren. *ATZ Automobiltech Z*, vol. 121, 2019.

[3] Liu-Henke X., Jacobitz S., Scherler S., Göllner M., Yarom O., Zhang J. A Holistic Methodology for Model-Based Design of Mechatronic Systems in Digitized and Connected System Environments. *16th International Conference on Software Technologies (ICSOFT)*, online, 2021.

[4] Lin Y.-C., Nguyen H.-L. T., Wang C.-H. Adaptive neuro-fuzzy predictive control for design of adaptive cruise control system. *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, Calabira, Italy, 2017.

[5] Abdullahi A., Akkaya S. Adaptive Cruise Control: A Model Reference Adaptive Control Approach. *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, Sinaia, Romania, 2020.

[6] Kiencke U., Nielsen L. *Automotive control systems: for engine driveline and vehicle*. Springer, Berlin, 2005.

[7] Shakouri P., Ordys A. Nonlinear model predictive control approach in design of adaptive cruise control with automated switching to cruise control. *Control Engineering Practice*, vol. 26, 2014.

[8] Anayor C., Gao W., Odekunle A. Cooperative Adaptive Cruise Control of A Mixture of Human-driven and Autonomous Vehicles. *SoutheastCon 2018*, St. Petersburg, FL, USA, 2018.

[9] Tirumala S.S. Evolving deep neural networks using coevolutionary algorithms with multi-population strategy. In: *Neural Comput & Applic*, vol. 32, 2020.

[10] Duriez T., Brunton S., Noack B. R. *Machine Learning Control*. Springer International Publishing, Cham, Switzerland, 2017.

[11] Lyu H. et. al. Esnet: Edge-Based Segmentation Network for Real-Time Semantic Segmentation in Traffic Scenes. *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019.

[12] Huang Z. et. al. Parameterized batch reinforcement learning for longitudinal control of autonomous land vehicles. *IEEE Trans. Syst., Man, Cybern, Syst.*, vol. 49, no. 4, 2019.

[13] Liu-Henke X. et. al. Holistic development of a full-active electric vehicle by means of a model-based systems engineering. *2016 IEEE International Symposium on Systems Engineering (ISSE)*, Edinburgh, UK, 2016.

[14] Yarom O. A. et. al. Artificial Neural Networks and Reinforcement Learning for model-based design of an automated vehicle guidance system. *12th International Conference on Agents and Artificial Intelligence (ICAART)*, Valletta, Malta, 2020.

[15] Rock J., Roth W., Toth M., Meissner P., Pernkopf F. Resource-Efficient Deep Neural Networks for Automotive Radar Interference Mitigation. in *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 4, 2021.

[16] Yarom O., Liu-Henke X. Development of a Simulation Environment for Automated Model Configuration for the Design and Validation of AI-Based Driving Functions. *11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*, online, 2021.