

## SEMINARARBEIT

Rahmenthema des Wissenschaftspropädeutischen Seminars:

*Bionik*

Leitfach: *Physik*

Thema der Arbeit:

### **Künstliche Neuronale Netzwerke**

und deren

### **Anwendung in einem Quadrocopter**

Verfasser: Benedikt Heyl

Kursleiter: OstR Müller

Abgabetermin:  
(2. Unterrichtstag im November)

8 November 2016

<b>Bewertung</b>	Note	Notenstufe in Worten	Punkte		Punkte
schriftliche Arbeit				x 3	
Abschlusspräsentation				x 1	
Summe:					
Gesamtleistung nach § 61 (7) GSO = Summe : 2 (gerundet)					

---

Datum und Unterschrift der Kursleiterin

# Inhaltsverzeichnis

<b>Inhaltsverzeichnis</b>	<b>1</b>
<b>1 Einleitung</b>	<b>2</b>
<b>2 Das Biologische Vorbild</b>	<b>3</b>
<b>3 Mathematische Umsetzung</b>	<b>4</b>
<b>4 Lernprozesse</b>	<b>6</b>
4.1 Details der Simulation	6
4.1.1 Abrundung der Aktivierungsfunktion	8
4.1.2 Verbesserung der Kostenfunktion	8
4.2 Evolutionäre Verbesserung	9
4.3 Backpropagation / Gradient Descent	9
<b>5 Praktische Umsetzung</b>	<b>12</b>
5.1 Problemstellung	12
5.2 Konventionelle Funktionsweise	14
5.3 Anwendung des künstlichen neuronalen Netzwerks	16
5.3.1 Ausrichtungswinkel	16
5.3.2 Sensordatenfusion durch das Neuronale Netzwerk	18
5.3.3 Verwendung von Rohdaten	18
5.3.4 Einführung eines Schubreglers	19
5.3 Vor und Nachteile von KNNs als Flugkontroll Software	20
<b>6 Ausblick</b>	<b>21</b>
<b>Abbildungsverzeichnis</b>	<b>23</b>
<b>Quellenverzeichnis:</b>	<b>24</b>
<b>Erklärung der Plagiatsfreiheit</b>	<b>25</b>

# 1 Einleitung

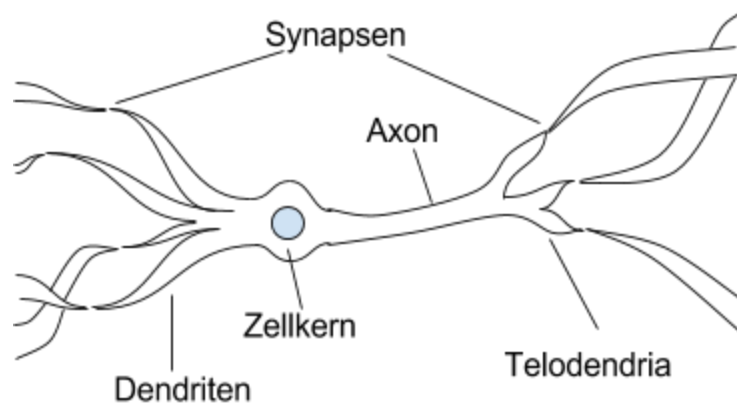
Computer sind großartig. Die Möglichkeit, Informationsverarbeitung an ein nicht menschliches System ab zu geben, war ohne Frage eine der, wenn nicht die größte technische Revolution.

Mit für ein menschliches Gehirn unerreichbarer Präzession und der Fähigkeit nichts zu vergessen haben Computer große Teile der Wirtschaft, der Wissenschaft, der zwischenmenschlichen Kommunikation und vieles mehr komplett verändert. Fast könnte man meinen, das menschliche Gehirn hätte seine Nützlichkeit verloren. Aber das ist nicht der Fall. Während sich Computer bei Aufgaben, die Präzision und Geschwindigkeit erfordern, sehr leicht tun, beispielsweise bei großen Rechenaufgaben oder dem Durchforsten von Datenbanken, sind Aufgaben, die weniger logisch aufgebaut sind für sie eine deutlich größere Herausforderung. Probleme wie etwa die bildbasierte Unterscheidung schien lange Zeit schier unlösbar. Auch Herausforderungen wie ein stabiler, aufrechter Gang, das Erkennen von Emotionen in einem Gesicht oder das kreative Erfinden von neuen Ideen, waren für einen Computer deutlich schwerer zu meistern als seinem feuchten Counterpart.

Deshalb übertrugen Informatiker abstrakte Konzepte des Gehirns in mathematische Modelle und waren damit in der Lage, es virtuell zu simulieren. Daraus entwickelte sich das mathematische Konzept der neuronalen Netzwerke. Es ist fast eine Unmöglichkeit, die daraus entstehenden Möglichkeiten zu überschätzen. Die Theorie entwickelte sich zwar schon in den späten achtziger Jahren des letzten Jahrhunderts, seit allerdings in den letzten Jahren moderne Rechenressourcen und die fast unbegrenzten Datenmengen des Internets zur Verfügung stehen, gelingen im Moment fast wöchentlich große Durchbrüche in in den Fähigkeiten von Computern. Neuronale Netzwerke sind in der Lage Annäherungs-entscheidungen zu treffen, die nicht auf harter Logik sondern auf Erfahrung aus Trainingsdaten basieren. Dies ermöglicht es Computern, Entscheidungen in der echten Welt zu treffen, die oft nicht sauber in Nullen und Einsen eingeteilt ist.

## 2 Das Biologische Vorbild

Das menschliche Gehirn und Nervensystem bestehen, genau wie bei jedem anderen Tier, aus einer sehr großen Zahl von Nervenzellen oder Neuronen. Diese Zellen sind für die Informationsverarbeitung und die Weiterleitung von Informationen im Körper zuständig. Ein Neuron hat wie jede andere Zelle einen Zellkern, der das genetische Material enthält und die Energieumsetzung der Zelle durchführt. Zusätzlich gibt es eine Reihe von Auswüchsen die die Kommunikation mit umliegenden Zellen ermöglichen. Ein Großteil dieser Verbindungen, genannt Dendriten, ist dafür zuständig Informationen zum Zellkern zu leiten und eine weitere, genannt Axon, ist für die Weitergabe zuständig. Das Axon verzweigt sich am Ende zu den so genannten Telodendria, und stellt an den Synapsen Kontakt zu einer Vielzahl anderer Zellen her<sup>1</sup>.



Abd. 1

Die Information, die zwischen den Neuronen ausgetauscht wird, wird mithilfe von sogenannten Aktionspotentialen weitergeleitet; Spannungsunterschieden im Millivoltbereich, die die Axone entlang wandern. Wenn die Information an eine andere Zelle weitergegeben wird, findet das an den Synapsen statt; ein schmaler Spalt zwischen Telodendria des präsynaptischen Neurons und dem Dendrit des postsynaptischen Neurons, der mit Neurotransmittermolekülen gefüllt ist. Die Synapsen spielen eine wichtige Rolle im Nervensystem, da sie entscheiden, ob und wie stark das

---

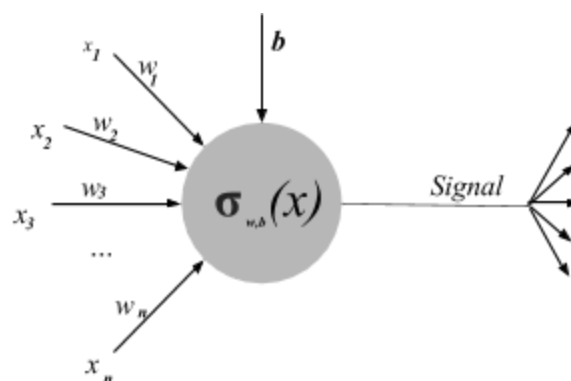
<sup>1</sup> Lexikon der Psychologie: Neuron

Signal an die postsynaptische Zelle weitergeleitet wird. Allerdings wird nicht jedes im Zellkern ankommende Signal über das Axon an alle angeschlossenen Zellen weitergeleitet. Nur wenn die Summe der ankommenden Signale über oder unter einem bestimmten Schwellenwert liegt “feuert” die Zelle.<sup>2</sup>

Dieses scheinbar einfache System kann erstaunliche Informationsverarbeitungsaufgaben bewältigen, wenn eine große Zahl von Neuronen in der richtigen Struktur angeordnet werden, die Neurotransmitter in den Synapsen richtig gewählt sind und die Neuronen die richtigen Schwellenwerte eingeprägt haben. Die Fähigkeiten der vielen einzelnen Neuronen entwickelt eine kollektive Komplexität,<sup>3</sup> die nicht nur in der Lage ist Informationen zu verarbeiten, sondern auch Erfahrungen zu speichern.

### 3 Mathematische Umsetzung

Um die Fähigkeiten der Nervenzellen in einer virtuellen Umgebung nutzen zu können, wurde nun eine Simulation der Fähigkeiten von Nervenzellen erstellt. Das einzelne Neuron wird damit zu einem virtuellen Objekt, das auf der einen Seite Zahlen aufnimmt und sie mit unterschiedlicher Gewichtung multipliziert (= Dendriten und Synapsen mit unterschiedlicher Kommunikationsstärke). Wenn die Summe dieser Inputs über einem bestimmten Schwellenwert liegt, wird ein Signal an andere Neuronen weitergeleitet.



Abd. 2

---

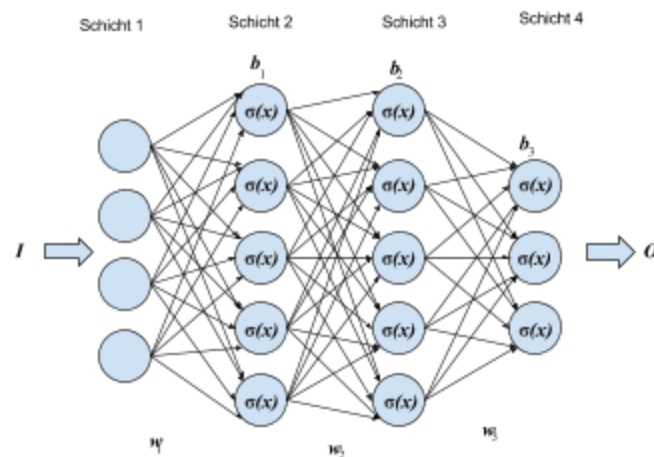
<sup>2</sup> Newmann, Section 5.2

<sup>3</sup> Gernshenson, 1

Diese Operation kann mathematisch abgebildet werden:

$$\sigma(x_i) = \begin{cases} 0 & \text{für } \sum_{i=0}^n x_i w_i \leq b \\ 1 & \text{für } \sum_{i=0}^n x_i w_i \geq b \end{cases}$$

Wobei  $x$  die Signale der präsynaptischen Zelle,  $b$  den Schwellenwert und  $w$  die Gewichtungen darstellt. Mit dieser vereinfachten Einheit eines Neurons, genannt Perzeptron, kann man nun ein sehr kleines künstliches Neuronales Netzwerk bauen.



Abd. 3

Dabei wird die ansonsten topologisch extrem komplizierte Struktur in einzelne sogenannte Schichten eingeteilt. Jede Schicht besteht aus einer gewissen Zahl von Neuronen, die Informationen von allen Neuronen der vorhergehenden Schicht empfangen und an alle Neuronen der nachfolgenden Schicht weitergeben. Schicht 1 nimmt die Information in Form einer Zahlenliste oder Vektors (im Beispiel vierdimensional) auf und gibt sie an Schicht 2 weiter. Schicht 2,3 und 4 multiplizieren die Signale mit den entsprechenden Gewichtungen und führen

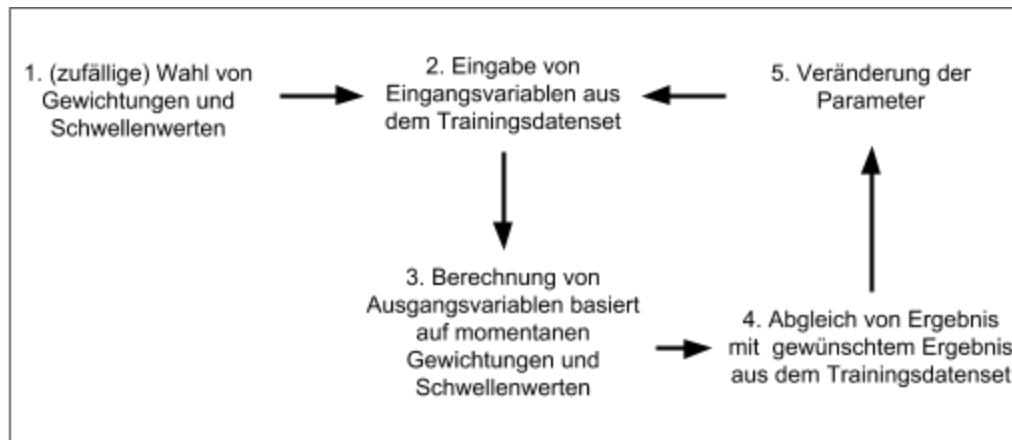
die Aktivierungsfunktion durch. Am Ende gibt Schicht 4 eine Zahlenliste oder Vektor (im Beispiel dreidimensional) aus. Die topologische Struktur ist nur durch die Anzahl der Dimensionen der Input- und Output-Vektoren beschränkt.

Die Stärke der Neuronalen Netzwerke liegt nun allerdings in ihrer Flexibilität. Jedes Problem, das ein Muster zwischen einem Inputvektor und einem Outputvektor finden und simulieren soll, kann durch Neuronale Netzwerke näherungsweise beschrieben werden.<sup>4</sup>

## 4 Lernprozesse

### 4.1 Details der Simulation

Die Methodik der Perzeptronen Netzwerke ist allerdings nur leistungsstark wenn alle Parameter richtig gesetzt sind. Um dieses Problem zu lösen sollte man sich das Netzwerk nur noch als mathematisches Konstrukt vorzustellen; als Gleichung, bei der die Input- und Output-Variablen Parameter und die Schwellenwerte und Gewichtungen Variablen sind. Die Parameter werden dann mit aus der Realität aufgezeichneten Trainingsdaten gefüllt und es wird eine sogenannte Kostenfunktion definiert, die in der Lage ist, den Unterschied des Outputs des künstlichen neuronalen Netzwerks (KNN) von seinem gewünschten Ergebnis zu beschreiben. Dann wird das Netzwerk nach folgender Methodik verbessert.



Abd. 4

---

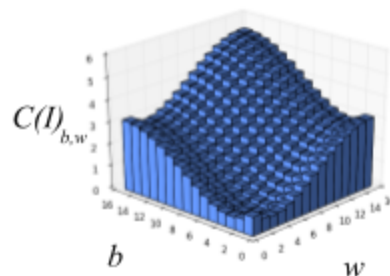
<sup>4</sup> Nielson, Chapter 4

Je nach dem welche Strategie gewählt wird, werden die Parameter im fünften Schritt unterschiedlich verändert. Die Kostenfunktion wird folgendermaßen definiert.

$$C(I) = \frac{\sum_{n=1}^m |Y_n - O(I)_n|}{m}$$

Wobei  $I$  die Eingangsvariablen,  $O$  die Ausgangsvariablen,  $Y$  das gewünschte Ergebnis und  $m$  die Anzahl der Beispiele im Trainingsdatensatz ist. Der Algorithmus geht also einmal durch den gesamten Datensatz durch und addiert für jede Ausgangsvariable den Unterschied zum gewünschten Ergebnis. Dann wird durch  $m$  geteilt, damit der Fehler unabhängig von der Länge des Datensatzes betrachtet werden kann. Vereinfacht auf 2 Variablen könnte die Kostenrechnung nun folgende Werte annehmen.

Diese Funktion gilt es nun zu minimieren, denn je kleiner die Kosten, desto näher ist das KNN am richtigen Ergebnis. Man kann es sich auch bildlich vorstellen: Der Algorithmus befindet sich immer an einer Stelle in diesem Diagramm und versucht mithilfe einer bestimmten Strategie die niedrigste Stelle zu finden. Allerdings muss für jeden Schwellenwert und jede Gewichtung eine zusätzliche Dimension eingeführt werden. Diese Visualisierung ist also, sobald mehr als zwei Variablen untersucht werden, nutzlos.



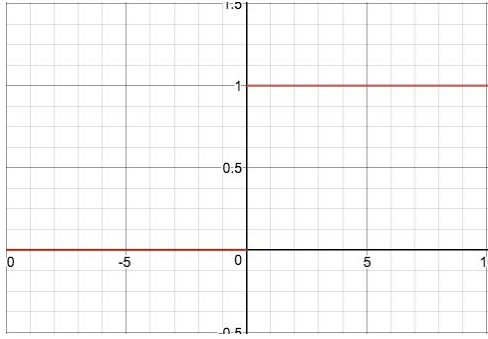
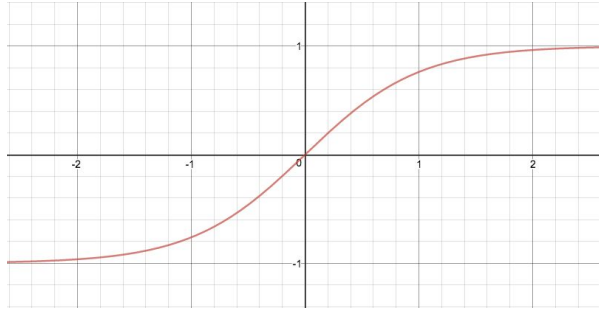
Abd. 5

Bevor im Folgenden nun zwei dafür anwendbare Strategien erläutert werden, kann der Vorgang noch mathematisch vereinfacht werden.



### 4.1.1 Abrundung der Aktivierungsfunktion

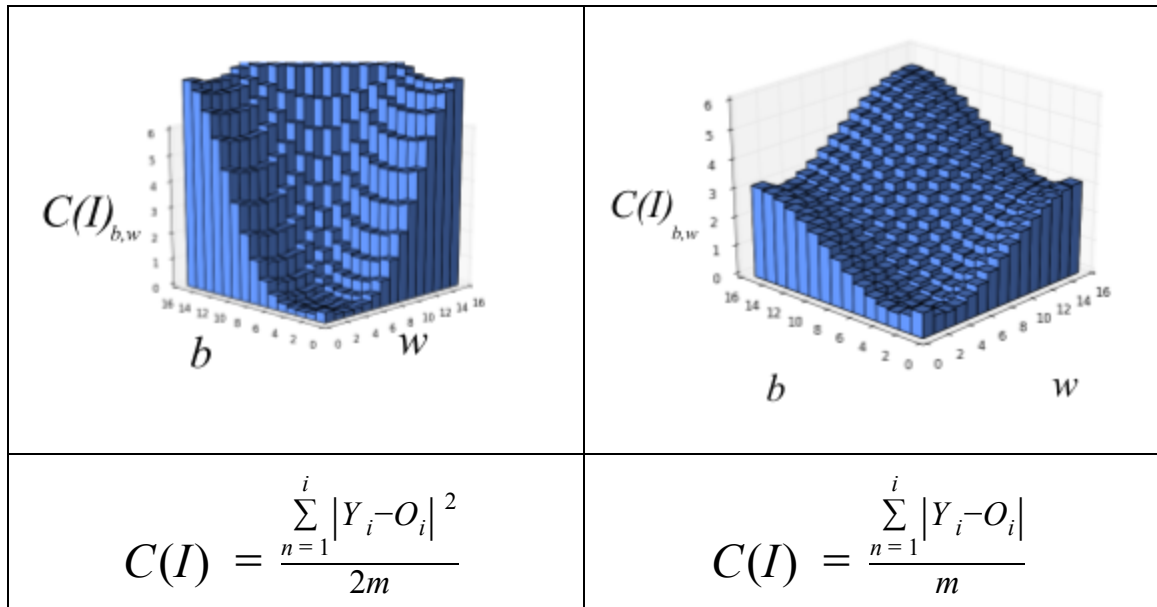
Wenn man bei einem Perzeptronen Netzwerk eine Variable oder einen Parameter ein bisschen verändert, verändert sich die Ausgangsvariable - aufgrund der Schwellenfunktion - entweder gar nicht oder sehr stark. Kleine Änderungen sind nicht möglich. Die Leistungsfähigkeit des KNNs verbessert sich also ungemein, wenn eine abgerundete Schwellenfunktion genutzt wird. Dabei gibt es eine Vielzahl von möglichen Funktionen (z.B. Sigmoid,  $\tan^{-1}(x)$ ...); alle mit ähnlicher Form. Auswahlkriterien können das gewünschte Lernverhalten oder die Berechnungsgeschwindigkeit auf der zur Verfügung stehenden Hardware sein.

	
Ursprüngliche Schwellenfunktion wenn $b = 0$	Hyperbolischer tangens wenn $b = 0$
$\sigma_b(x_i) = \begin{cases} 0 & \text{für } \sum_i x_i w_i \leq b \\ 1 & \text{für } \sum_i x_i w_i \geq b \end{cases}$	$\sigma_b(x_i) = \tanh(x + b)$

Abd. 6

### 4.1.2 Verbesserung der Kostenfunktion

Quadratische Funktionen weisen ein extremeres Steigungsverhalten als lineare Funktionen aus. Deswegen ist es günstiger, bei der Kostenrechnung den Unterschied erst zu quadrieren und dann durch zwei zu teilen, da Minima im Graphen dann klarer zu erkennen sind.



Abd. 7

## 4.2 Evolutionäre Verbesserung

Diese Methode ist wohl die naheliegendste und orientiert sich an der Entwicklung des Gehirns. Der Algorithmus probiert einen Schwarm verschiedener Neuronaler Netzwerke aus, berechnet, welche die beste Leistung aufweisen, kombiniert deren Merkmale und führt kleine zufällige Veränderungen ein. Ähnlich wie in der Natur setzt sich dann eine evolutionäre Verbesserung ein, die sich langsam an ein Ideal hinarbeitet. Diese Methode ist zwar einfach umzusetzen, verursacht allerdings einen enormen Rechenaufwand, der mit der Anzahl der Variablen exponentiell ansteigt.

## 4.3 Backpropagation / Gradient Descent

Eine Funktion zu minimieren ist allerdings auch nur ein mathematisches Problem, das heißt, man kann es mit mathematischen Methoden lösen. Vereinfacht könnte man den Backpropagation Algorithmus folgendermaßen erklären: Man leitet die Kostenfunktion für jede einzelne Variable ab und findet damit heraus, was eine kleine Änderung der Variable an der Leistungsfähigkeit des KNNs verändern würde. Dann verändert man die Variablen ein kleines bisschen, um die Kostenfunktion zu verkleinern und die Leistungsfähigkeit zu verbessern. Visuell kann man sich

vorstellen, dass der Algorithmus an jedem Punkt mithilfe der Ableitung feststellen kann in welche Richtung es auf dem Diagramm Berg ab geht, und macht in diese Richtung einen Schritt.

Um die Kostenfunktion abzuleiten ist es nötig, die Operationen des KNNs mithilfe von Matrizen abzubilden. Der Prozess der Weitergabe mit verschiedener Gewichtung stellt sich praktischerweise als ein Skalarprodukt mit einer Gewichtsmatrix heraus. Ein Netzwerk mit vier Schichten wie oben abgebildet, kann dann mit folgender Gleichung beschrieben werden:

$$O(I) = \tanh(b_3 + w_3 \tanh(b_2 + w_2 \tanh(b_1 + w_1 I)))$$

Wobei  $\tanh(x)$  die Aktivierungsfunktion,  $I$  die Eingabevariablen,  $b_{1,2,3}$  die Schwellenwertvektoren mit so vielen Dimensionen wie es Neuronen in der Schicht gibt und  $w_{1,2,3}$  die Gewichtungsmatrizen mit so vielen Reihen wie es Neuronen in dieser Schicht gibt und so vielen Spalten wie es Neuronen in der vorangegangenen Schicht gibt. Vereinfacht:

$$\begin{aligned} O(I) &= \tanh(z_L) = a_L \\ a_l &= \tanh(b_l + w_l a_{l-1}) = \tanh(z_l) \end{aligned}$$

Wobei  $L$  die letzte Schicht,  $l$  eine beliebige Schicht beschreibt. Berechnet man nun die Ableitung oder den Gradienten für die Kostenfunktion ( $\nabla C$ ) kann man die Veränderung der Leistungsfähigkeit des KNNs für kleine Parameterveränderungen abschätzen<sup>5</sup>.

$$\Delta C \approx \nabla C * \Delta p$$

Wobei  $\Delta p$  die Parameterveränderung darstellt. Nun wird  $\Delta p$  folgendermaßen definiert:

---

<sup>5</sup> Nielson, Chapter 1

$$\Delta p = -\mu \nabla C$$

Wobei  $\mu$  eine kleine positive Konstante, genannt Learning Rate darstellt. Wenn die Learning Rate klein genug gewählt wird ist  $\Delta C$  nun zwingend negativ und  $C(I)$  wird minimiert da:

$$\Delta C \approx \nabla C * -\mu \nabla C$$

$$\frac{\Delta C}{(\mu * (\nabla C^2))} \approx -1$$

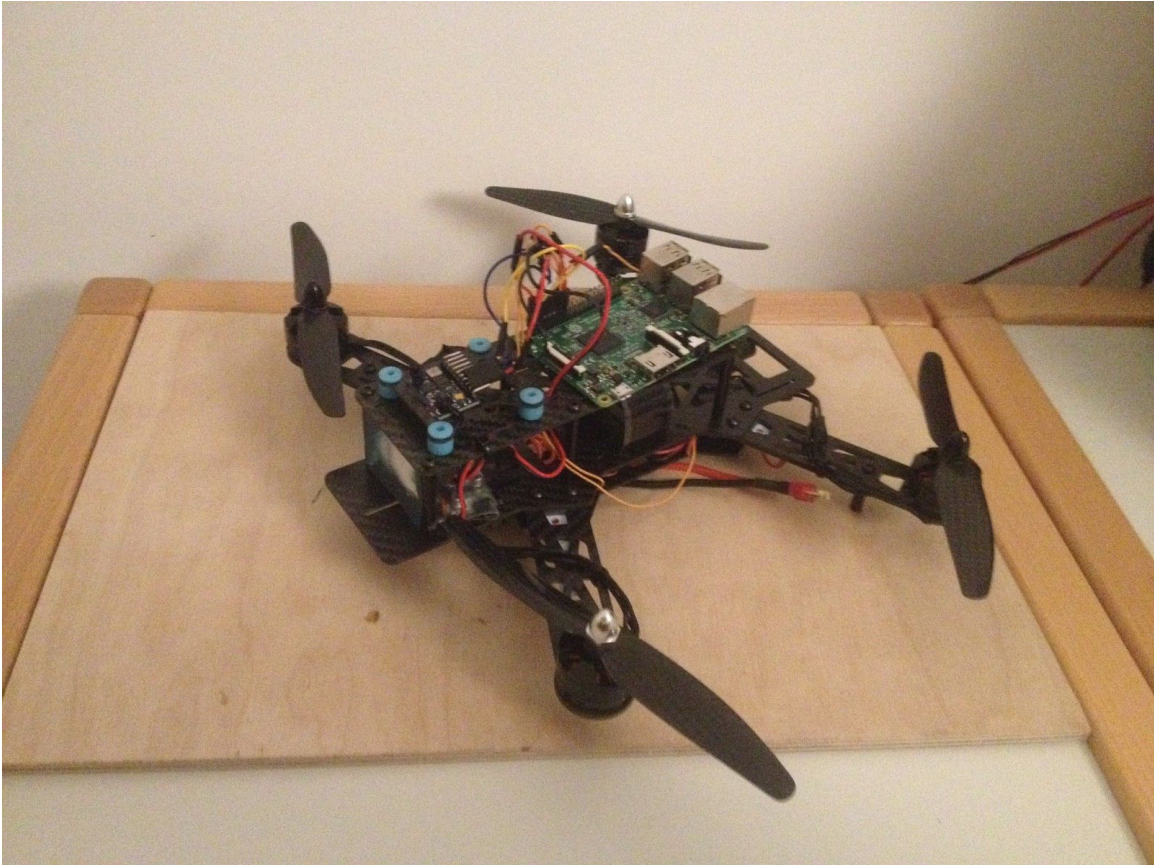
Nun wird jedem Parameter  $p$  folgender Wert zugewiesen:

$$p \rightarrow \bar{p} = p - \mu \frac{\partial C}{\partial p}$$

Die genaue Herleitung von  $\nabla C$  befindet sich im Anhang, aber wichtig um die Vorteile von Backpropagation zu verstehen, sind folgende Punkte.

- Wie oben bewiesen, tritt bei einer adäquaten Learning Rate bei jeder Wiederholung eine Verbesserung ein.
- Um die Ableitung zu berechnen arbeitet man sich rückwärts durch das Netzwerk
- Große Teile der Ableitung können dank der Kettenregel immer wieder wiederholt werden.

Der letzte Punkt ist der große Vorteil des Backpropagation Algorithmus, der den Rechenaufwand deutlich einschränkt, da dieser mit der Vergrößerung des Netzwerks nicht exponentiell, sondern linear anwächst.



Abd. 8

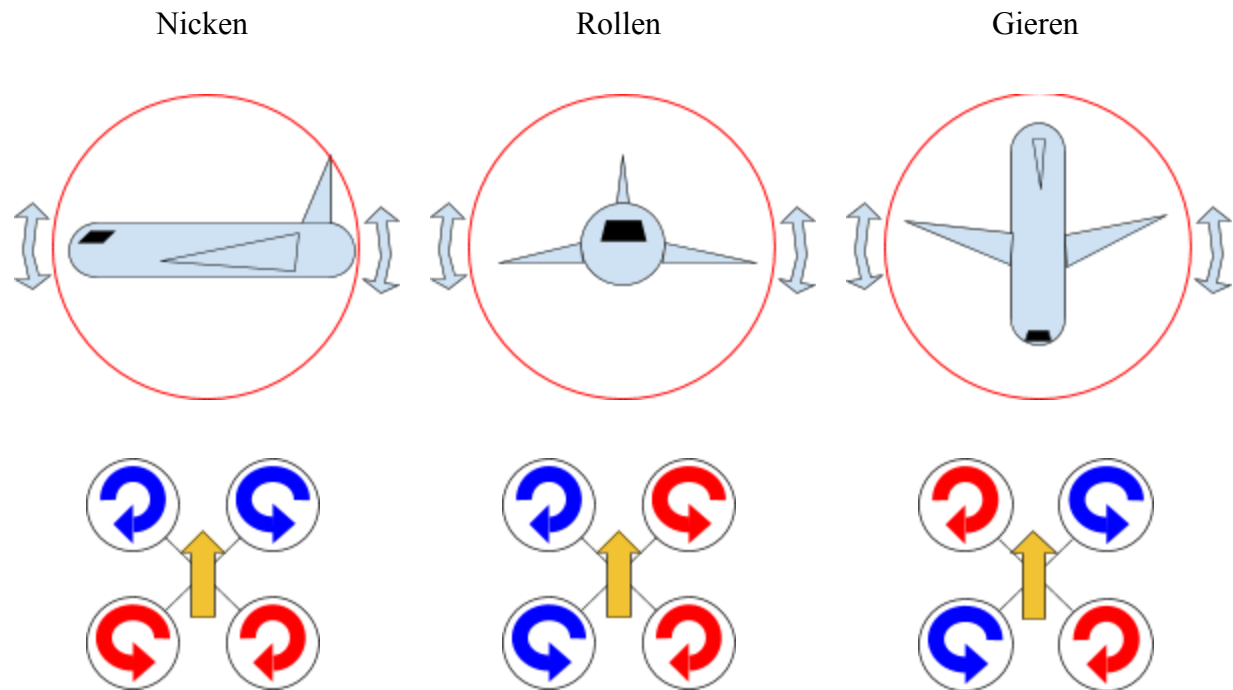
## 5 Praktische Umsetzung

Um eine mögliche Verwendung eines KNNs zu demonstrieren, soll dieses nun dazu genutzt werden einen selbstgebaute Quadcopter stabil in der Luft schweben zu lassen. Ein Quadcopter ist ein Fluggerät mit vier symmetrisch angeordneten Propellern. Eine genaue Beschreibung des Systems findet sich im Anhang.

### 5.1 Problemstellung

Die Orientierung eines Luftfahrzeugs im Raum kann durch drei Drehachsen beschrieben werden. Die Aufgabe des Flug Computers (FC) besteht darin, die Ausrichtungswinkel der Drone möglichst nahe an die gewünschte Ausrichtung zu bringen.

Bei einem Flugzeug gibt es für jeden dieser Drehachsen mindestens eine designierte Leitfläche und der Vortrieb wird von einem separaten Antrieb gewährleistet. Ein Quadrocopter muss die selben drei Rotationsachsen und den Vortrieb mit nur vier Rotoren kontrollieren.



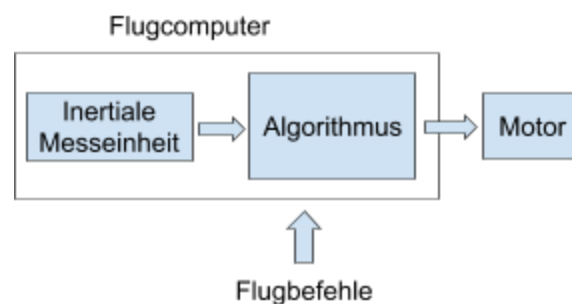
Abd. 9

Rollen wird erreicht indem die Drehzahl der rechten Rotoren sich von der der linken unterscheidet und Nicken durch ein Drehzahlunterschied zwischen vorne und hinten. Da sich die Hälfte der Rotoren im Uhrzeigersinn und die andere gegen den Uhrzeigersinn drehen muss, um den Drehimpuls im System aus zu gleichen, erreicht ein Drehzahlunterschied zwischen den im Uhrzeigersinn drehenden Motoren und den gegen den Uhrzeigersinn drehenden Motoren eine Gierbewegung. Vortrieb wird erzeugt, in dem der Quadrocopter eine konstant schräge Position hält und damit seinem Gesamtschubvektor neben der vertikale auch ein horizontale Komponente hinzufügt.

Da die Anzahl der Freiheitsgrade größer ist als die der Kontrolleinflüsse nennt man das Fluggerät unteraktuiert, was die Kontrolle des Systems erschwert. Zusätzlich besitzt ein Quadrocopter, anders als ein Flugzeug, keine Eigenstabilität. Ein Flugzeug tendiert, wenn es keine Kontrolleinflüsse erfährt zu einem stabilen Zustand, d.h. wenn ein Flugzeug ohne Inputs durch einen Piloten oder einen Computer geradeaus fliegt, wird es weiterhin geradeaus fliegen auch wenn Störeinflüsse (z.B. Wind) von außen auf das System einwirken. Diese Stabilität liegt bei Quadrocoptern nicht vor. Ein Computer muss kontinuierlich störenden Umwelteinflüssen entgegenwirken damit die Drohne nicht kippt oder abstürzt.

## 5.2 Konventionelle Funktionsweise

Der FC erhält Informationen über den Zustand der Drohne durch Kommunikation mit der Inertialen Messeinheit. Dort befinden sich zwei Messgeräte: ein Beschleunigungssensor, der Beschleunigung in allen drei Raumdimensionen misst und ein Gyroskop, das Rotationsgeschwindigkeiten in allen drei Rotationsachsen sammelt. Um einen stabilen Flug zu erreichen, liest der FC mindestens 100 mal pro Sekunde die Daten von den Sensoren und berechnet, welche Drehzahl der Motoren die Flugbefehle erfüllen würden.



Abd. 10

Normalerweise geschieht dies durch zwei Schritte: Zuerst müssen die Ausrichtungswinkel berechnet werden. In unserem Fall entspricht das dem *arctan* der relevanten Beschleunigungsvektoren und den über die Zeit integrierten Rotationsgeschwindigkeitsmessungen. Diese beiden Werte werden nun nach einer bestimmten Formel kombiniert, in unserem Fall mit einem gewichteten Durchschnitt.

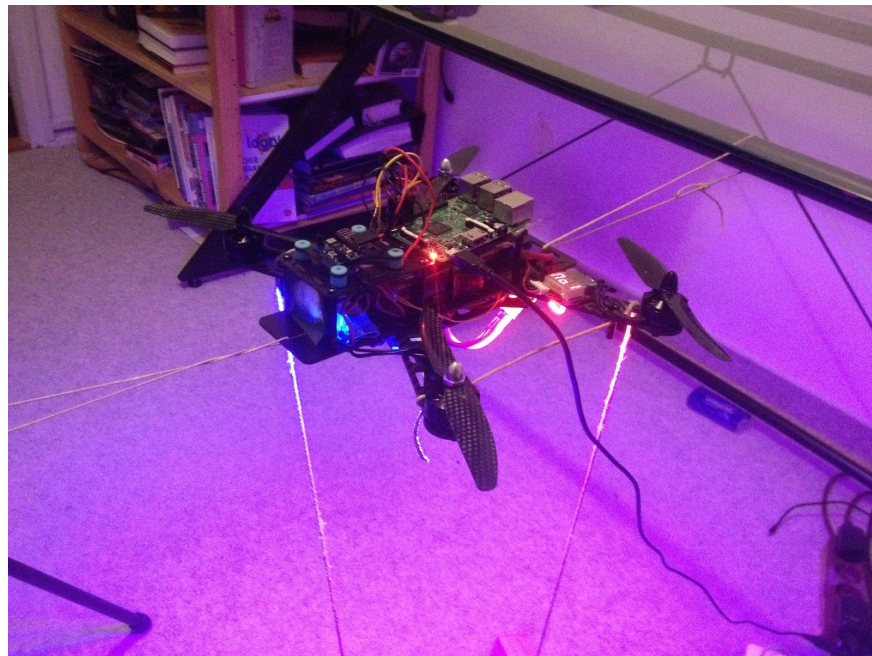


Die Schubkraft  $S$  zum Zeitpunkt  $t$  eines Motors wird dann folgendermaßen berechnet:

$$S_t(B, D, S_{t-1}) = B \pm (K_p D - K_I(S_{t-1} - B) + K_D \frac{dD}{dt})$$

Wobei  $D$  die Abweichung vom gewünschten Ausrichtungswinkel,  $S_{t-1}$  die Schubkraft zum vorhergegangenen Zeitpunkt,  $B$  die vom Kontrollbefehl geforderte Schubkraft und  $\frac{dD}{dt}$  die Ableitung des Ausrichtungswinkels über die Zeit (d.h die Rotationsgeschwindigkeit) darstellt.  $K_p$ ,  $K_I$  und  $K_D$  beschreiben systemspezifisch gewählte, positive Konstanten, die vom Entwickler festgelegt werden müssen.

Wenn man das Problem vereinfacht, ist es allerdings auch nur eine Funktion, die aus einem Vektor von Sensordaten und Flugbefehlen einen Vektor von Motordaten erstellt und deswegen ohne Probleme mit einem Neuronalen Netzwerk annäherbar. Diese Alternative soll im folgenden erkundet werden.



Abd. 11



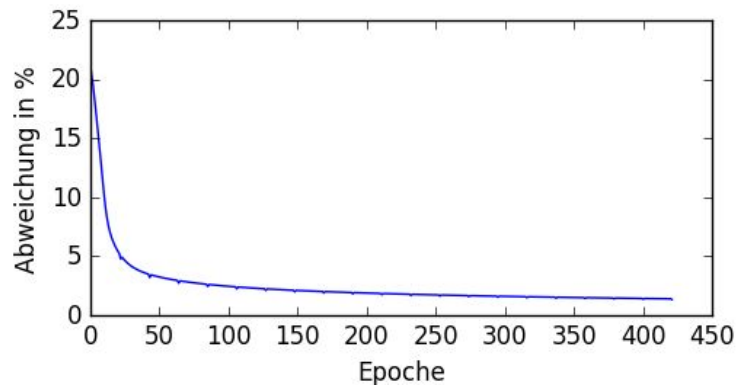
## **5.3 Anwendung des künstlichen neuronalen Netzwerks**

Um die Lösung des Problems zu vereinfachen wurde die Drohne, wie in Abbildung 11 sichtbar, mithilfe eines Seils so fixiert, dass nur eine Bewegung in einer Rotationsachse stattfinden konnte. Zusätzlich lag der Fokus nur auf dem stabilen Halten eines geraden Zustands und es wurden fast keine Flugbefehle an den FC abgegeben. Dann wurde in drei Stufen mehr und mehr der Kontrolle Autorität an KNNs abgegeben.

### **5.3.1 Ausrichtungswinkel**

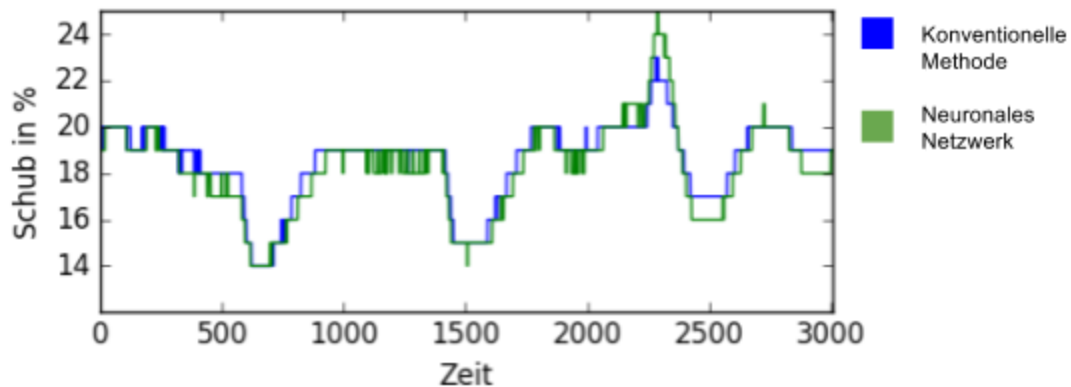
Die Drohne wurde basiert auf einem konventionellen Algorithmus, stabilisiert; während ca. Drei Minuten stabilen Flugs wurden dann die berechneten Ausrichtungswinkel, die Rotationsgeschwindigkeit und die Steuerbefehle an die Motoren in Prozent aufgezeichnet. Dies entspricht 40000 Kontroll Iterationen und etwa 2MB in Binärdaten.

Nun wurde ein KNN darauf trainiert, aus den Messdaten die richtigen Steuerbefehle für die Motoren zu berechnen. Das hier benutzte KNN weist zwei Neuronen in der Inputschicht, einen für den Ausrichtungswinkel und einen für die Rotationsgeschwindigkeit auf. Dann folgen zwei Schichten von je vier Neuronen und die Output Schicht, ebenfalls mit vier Neuronen, die jeweils einem Motor zugeordnet werden. Um den Lernprozess zu vereinfachen wurde der gesamte Trainingsdatensatz auf den Bereich zwischen  $-1$  und  $1$  skaliert, da die Aktivierungsfunktion nur in diesem Bereich sinnvolle Werte ausgibt. Mithilfe des in Kapitel 4.3 beschriebenen Backpropagation Algorithmus wurden dann das Netzwerk trainiert. Alle Messdaten wurden nacheinander in den Algorithmus eingeführt und ein gewisser Output produziert, der von dem gewünschten Output abweicht. Die durch die Kostenfunktion gemessene "Fehlerhaftigkeit" des erhaltenen Outputs wird dann mit einer Learning Rate von  $0.08$  minimiert.



Abd. 12

Nach etwa drei Minuten Training und ca. 400 Epochen war der Quadrocopter in der Lage mit sehr hoher Präzision die Kontrolleinflüsse des konventionellen Algorithmus nachzuahmen. In der untenstehenden Graphik wurde während einem kurzen Zeitraum die Ergebnisse beider Algorithmen aufgezeichnet. Eine klare Korrelation zwischen den beiden Ergebnissen ist zu erkennen.



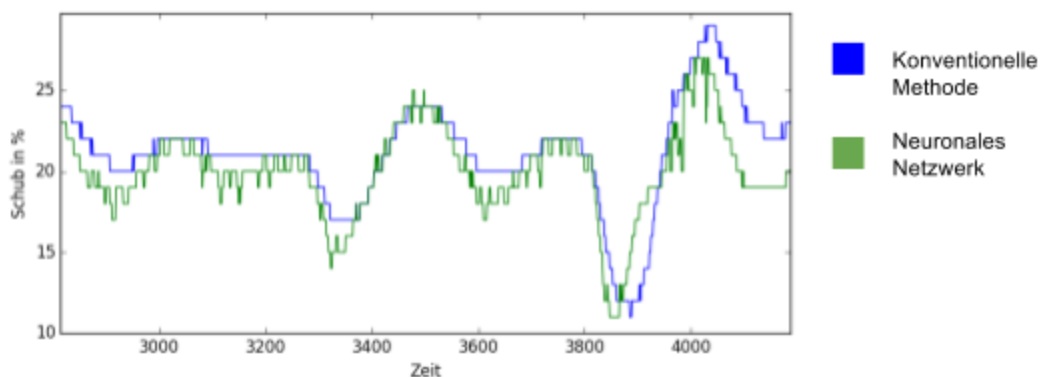
Abd. 13

Das neuronale Netzwerk stabilisierte an diesem Punkt erfolgreich die Ausrichtung der Drohne und reagierte angemessen auf Störeinflüsse.

### 5.3.2 Sensordatenfusion durch das Neuronale Netzwerk

Eine Komplexität in der Entwicklung der Software für einen FC liegt in der Sensordatenfusion, also in welchen Situationen sich der Flugcomputer auf welchen Sensor verlässt. Anders als in diesem Experiment, wird in vielen Quadrocoptern komplizierte Algorithmen wie zum Beispiel der Kalman Filter angewendet.<sup>6</sup> Bei der Nutzung von neuronalen Netzwerken liegt allerdings die Weitergabe dieses Problems an das Netzwerk nahe.

Dafür wurde nun der Input Datenvektor des Systems vergrößert und anstatt der komplett berechneten Ausrichtungswinkel wurden die einzeln berechneten Ausrichtungswinkel der verschiedenen Sensoren in das neuronale Netzwerk eingeführt. Das KNN lernt dann selbstständig auf welchen Sensor in welchem Fall Verlass ist. Da das Netzwerk immer noch auf der Basis konventioneller Telemetriedaten trainiert wird, kann es keinen besseren Sensorfusionsalgorithmus entwickeln als jener, der im konventionellen System eingesetzt wird, allerdings wird der Rechenaufwand des FC reduziert. Auch mit dieser Methode konnte erfolgreich eine stabiler Flug erzeugt werden.



Abd. 14

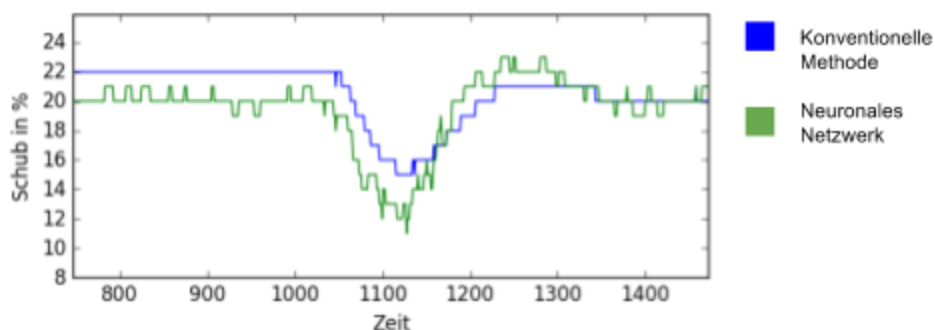
### 5.3.3 Verwendung von Rohdaten

Der letzte Entwicklungsschritt kann allerdings noch weiterentwickelt werden. Die Input Daten sind immer noch nicht unbehandelt, da die Sensoren nicht Ausrichtungswinkel sondern

---

<sup>6</sup> Olejnik, Peter S.49

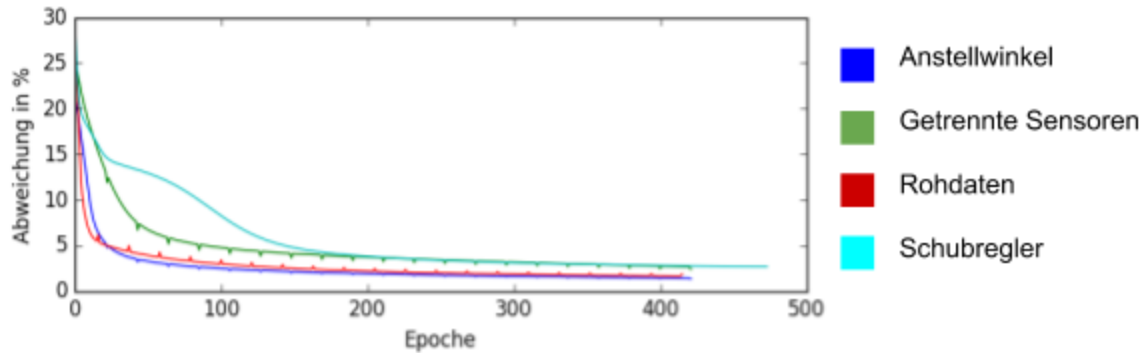
Beschleunigungsvektoren und Rotationsgeschwindigkeiten messen. Folglich wurde das KNN modifiziert, um mit genau diesen Rohdaten umzugehen. Der Inputvektor besteht nun also aus der Vertikalbeschleunigung, der Horizontalbeschleunigung und der Rotationsgeschwindigkeit. Bei diesem Entwicklungsschritt ist es wichtig zu beachten, dass der FC mit einer konstanten Kontrollfrequenz operiert, da sonst die Rotationsgeschwindigkeitsmessungen von dem neuronalen Netzwerk nicht sinnvoll über die Zeit integriert werden können. Nichtsdestotrotz wurde die Drohne erfolgreich von einem solchen KNN gesteuert.



Abd. 15

### 5.3.4 Einführung eines Schubreglers

Um dem KNN nun die Fähigkeit zu geben auf Steuerbefehle von außen ein zu gehen, wird eine weitere Variable, der Schub eingeführt. Während der Quadrocopter Daten sammelt, geht er immer von einem gewissen Schub aus, der dem Quadrocopter von außen vorgegeben wird. In den vorangegangenen Experimenten war dieser Schub immer konstant bei 20%. Um nun die verschiedenen Flugverhalten bei verschiedenen Schubeinstellungen zu erkunden, wird während der Aufzeichnungsphase der Schub zwischen 10% und 30% variiert und der Wert zu jedem Zeitpunkt aufgezeichnet. Das KNN wird nun um eine weitere Inputzelle erweitert, in der der Schubwert angegeben wird. Wenn man die Lernverhalten aller dieser Methoden vergleicht, kann man klar feststellen, dass die Verwendung von Schubdaten den Lernprozess deutlich verlängert.



Abd. 16

Nichtsdestotrotz kann nach der Trainingsphase der Schub verändert werden und der FC verändert den Betrag des Gesamtschubvektors adäquat. Auch an diesem Punkt bleibt die Drohne stabil in einer waagrechten Position.

### 5.3 Vor und Nachteile von KNNs als Flugkontroll Software

Die in dieser Arbeit beschriebenen Kontrollalgorithmen sind in der Lage, dieselben Operationen, die ein konventioneller Algorithmus ausführt mit wesentlich geringerem Rechenaufwand durchzuführen. Bei einer kleinen Drohne, die nur eine sehr begrenzte Energiekapazität in der Batterie zur Verfügung stehen hat, kann diese geringe Energieersparnis zu deutlich verlängerten Flugzeiten führen.

Allerdings ist der Algorithmus immer nur in der Lage den konventionellen Algorithmus nachzuahmen und wird nie eine präzisere Kontrolle gewährleisten. Die Anwendung von sogenannten Reinforcement Learning Strategien, die ebenfalls auf KNNs basieren, würde es ermöglichen, dass FC mit der Zeit immer mehr über das System lernt und immer genauere Kontrolle ausübt.

Ein Nachteil bei der Nutzung von KNNs ist allerdings, dass der Algorithmus eine black box darstellt und es unmöglich ist zu wissen, warum er funktioniert und wichtiger, wo die Grenzen seiner Fähigkeit liegen. Dies könnte bei der Benutzung in einem FC ein Sicherheitsrisiko darstellen.

## 6 Ausblick

Das im letzten Kapitel beschriebene System kann lediglich als ein kleiner Einblick in das Potential von künstlichen Neuronalen Netzwerken gesehen werden. Aus der flexiblen Struktur des Algorithmus ergibt sich eine große Zahl von möglichen Anwendungen.

Ein großes Feld, das schon heutzutage diese Technologie nutzt, ist der Bereich der Computer-Nutzer Interaktion. Oft ist diese von Frustration geprägt, da der Computer eine geringe Fehlertoleranz aufweist und sich nicht an den Kontext anpassen kann. Diese Fähigkeiten in ein traditionelles System einzuarbeiten würde einen enormen Entwicklungsaufwand erfordern. KNNs sind allerdings in der Lage, mit diesen Ungenauigkeiten umzugehen und sich flexibel an ihre Umgebung anzupassen. Deswegen werden sie schon heute in Sprachassistenten <sup>7</sup> und automatischen Übersetzungsservices <sup>8</sup> genutzt.

Es entstehen allerdings auch eine große Zahl von neuen Nutzungsfeldern, bei denen Computer nun in der Lage sind, mit Problemen umzugehen, die vorher nur mit menschlicher Hilfe gelöst werden konnten. Beispielsweise können mit sogenannten Convolutional Neural Nets bildgebende medizinische Untersuchungsmethoden analysiert werden.<sup>9</sup> Auch die Fähigkeit, die Entwicklung des Aktienmarktes abzuschätzen wurde schon demonstriert.<sup>10</sup>

---

<sup>7</sup> Beaufays, Françoise

<sup>8</sup> Le, Quoc V. und Mike Schuster

<sup>9</sup> Preis, Ori, Michael A. Blake und James A.Scott

<sup>10</sup> u.a. Kshirsagar, Gaurav

Außerdem profitiert natürlich auch das Feld der Robotik -wie auch in dieser Arbeit bewiesen - von KNNs. Sie bringen einem robotischen Arm selbstständig bei, wie man eine Tür öffnet <sup>11</sup> und erlauben selbstfahrenden Autos die Kamerabilder seiner Umwelt zu verstehen <sup>12</sup>.

Künstliche neuronale Netzwerke haben noch lange nicht alles erreicht, was erreicht werden kann und einige der größten technologischen Durchbrüche der Zukunft werden sehr wahrscheinlich auf dieser Technologie basieren.

---

<sup>11</sup> Levine, Sergey

<sup>12</sup> The Tesla Team

## Abbildungsverzeichnis

- 1: Der Aufbau einer Nervenzelle
- 2: Die Schematik eines virtuellen Neurons
- 3: Ein Perzeptronennetzwerk mit vier Schichten
- 4: Der rekursive Verbesserungsprozess eines künstlichen Neuronalen Netzwerks
- 5: Ein möglicher Graph für die Werte einer Kostenfunktion  $C$
- 6: Vergleich zweier möglicher Aktivierungsfunktionen
- 7: Der Vergleich von quadratischer und linearer Kostenfunktion
- 8: Der im Experiment behandelte Quadrocopter
- 9: Drehwinkel bei Luftfahrzeugen und deren Umsetzung bei einem Quadrocopter
- 10: Der Aufgaben eines Flugcomputers
- 11: Dieser Aufbau beschränkt die Rotation des Quadrocopters
- 12: Lernverhalten des Backpropagationalgorithmus eines auf Anstellwinkel trainiertes KNN
- 13: Vergleich von konventionellem Algorithmus und ein durch den Anstellwinkel trainiertes KNN
- 14: Vergleich von konventionellem Algorithmus und ein durch die Messungen der einzelnen Sensoren trainiertes KNN
- 15: Vergleich von konventionellem Algorithmus und ein durch Rohdaten trainiertes KNN
- 16: Vergleich der Lernverhalten der verschiedenen demonstrierten Methoden



## Quellenverzeichnis:

- [1] "Lexikon der psychologie: Neuron", Spektrum, 2000, Abgerufen: 2.8.2016, <http://www.spektrum.de/lexikon/psychologie/neuron/10516>
- [2] Newmann, Mark E.J., "Networks: An Introduction", Oxford: Oxford University Press, 2010, Section 5.2, Abgerufen 25.10.2016, <https://books.google.de/books?id=LrFaU4XCsUoC&printsec=frontcover&dq=Networks+an+Introduction&hl=en&sa=X&ved=0ahUKewjX582xpqjNAhXiNJoKHawtALoQ6AEIHDAA#v=onepage&q&f=true>
- [3] Gernshenson, Carlos, "Artificial Neural Networks.", Arxiv: Neural and Evolutionary Computing, 2003, Abgerufen: 2.10.2016, <https://arxiv.org/abs/cs/0308031>
- [4,5] Nielsen, Michael A., "Neural Networks and Deep Learning", Determination Press 2015, Abgerufen: 5.11.2016, <http://neuralnetworksanddeeplearning.com/index.html>
- [6] Olejnik, Peter U., "Design and Implementation of a Controller for a BeagleBone Quadcopter", Graduate Thesis - Mechanical Engineering 2016, Abgerufen: 5.11.2016, [http://scholar.rose-hulman.edu/cgi/viewcontent.cgi?article=1008&context=mechanical\\_engineering\\_grad\\_theses](http://scholar.rose-hulman.edu/cgi/viewcontent.cgi?article=1008&context=mechanical_engineering_grad_theses)
- [7] Beaufays, Françoise, "The neural networks behind google voice", Google Research Blog 2015, Abgerufen: 2.10.2016, <https://research.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html>
- [8] Le, Quoc V./Schuster, Mike, "A Neural Network for Machine Translation, at Production Scale", Google Research Blog 2016, Abgerufen am 2.10.2016, <https://research.googleblog.com/2016/09/a-neural-network-for-machine.html>;
- [9] Preis, Ori/Blake, Michael A. /Scott, James A., "Neural Network Evaluation of PET Scans of the Liver: A Potentially Useful Adjunct in Clinical Interpretation", Radiology (*Volume 258, Issue 3*) 2011, Abgerufen: 2.10.2016, <http://pubs.rsna.org/doi/full/10.1148/radiol.10100547>
- [10] u.a. Kshirsagar, Gaurav, "Stock Market Prediction using Artificial Neural Networks", International Journal of Advanced Research in Computer Engineering & Technology (*Volume 5 Issue 5*) 2016, Abgerufen: 5.10.2016, <http://ijarcet.org/wp-content/uploads/IJARCET-VOL-5-ISSUE-5-1691-1695.pdf>
- [11] Levine, Sergey, "Deep Learning for Robots: Learning from Large scale Interaction", Google Research Blog 2016, Abgerufen: 6.10.2016, <https://research.googleblog.com/2016/03/deep-learning-for-robots-learning-from.html>
- [12] The Tesla Team, "All Tesla Cars being Produced Now Have Full Self-Driving Hardware", Tesla Motors Blog 2016, Abgerufen: 19.10.2016, <https://www.tesla.com/blog/all-tesla-cars-being-produced-now-have-full-self-driving-hardware>

## **Erklärung der Plagiatsfreiheit**

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen (einschließlich elektronischer Quellen und dem Internet) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht.

---

Datum Name, Vorname

Unterschrift