Berufsmaturitätsarbeit 2019-2020

Hardware-Neuron

Tendai Rondof / Malik Schneider







Inhalt

[Vorwort 4](#_Toc31016536)

[Tendai 4](#_Toc31016537)

[Malik 4](#_Toc31016538)

[Danksagungen 4](#_Toc31016539)

[Abstract 4](#_Toc31016540)

[Einleitung 4](#_Toc31016541)

[Hauptteil 5](#_Toc31016542)

[Theorie 5](#_Toc31016543)

[Grundlage der Problematik 5](#_Toc31016544)

[Test Bedingungen 5](#_Toc31016545)

[Möglichkeiten 7](#_Toc31016546)

[Unmöglichkeiten 7](#_Toc31016547)

[Bedingungen/Einschränkungen 7](#_Toc31016548)

[Typ des Perceptrons (Vorteile gegenüber anderen Methoden) 7](#_Toc31016549)

[Mathematischer Ansatz vorwärts Rechnung 7](#_Toc31016550)

[Aktivierung 8](#_Toc31016551)

[Fehlerberechnung 8](#_Toc31016552)

[Problematik von Theorie und Praxis 9](#_Toc31016553)

[Adaption von der Theorie zur Machbarkeit 9](#_Toc31016554)

[Neue Bedingungen und Nachteile 9](#_Toc31016555)

[Schema Blockschaltbild 9](#_Toc31016556)

[Detailiertes Gesamtschema von EAGLE 10](#_Toc31016557)

[Erklärung der Funktion 11](#_Toc31016558)

[Ganzes Neuron 11](#_Toc31016559)

[Input 11](#_Toc31016560)

[DeltaApply 12](#_Toc31016561)

[Core 12](#_Toc31016562)

[Activation 13](#_Toc31016563)

[Vergleicher OPV 14](#_Toc31016564)

[DeltaCalc 14](#_Toc31016565)

[Layout/Bestückungsplan von EAGLE 16](#_Toc31016566)

[Stückliste 17](#_Toc31016567)

[Messungen 18](#_Toc31016568)

[Blockschaltbild des Versuchsaufbaus 18](#_Toc31016569)

[Programmierung des Mikrokontrollers 18](#_Toc31016570)

[Benutzeranleitung 18](#_Toc31016571)

[Abkürzungsverzeichnis 18](#_Toc31016572)

# Vorwort

Seit einiger Zeit boomt das Interesse an künstlichen Intelligenzen und neue Ideen, wie sie umgesetzt werden können, sind hochgefragt. Wir erwarten nicht dass wir mit einem Analogen Neuron das Rad neu erfinden, jedoch gelingt es uns einen Weg zu beschreiten der bis jetzt noch niemand gedacht hat.

## Tendai

Ich habe dieses Thema ausgewählt, weil Künstliche Intelligenzen mich seit Jahren interessieren. Durch dieses Projekt kann ich den Wunsch ein Neuron auf Hardwarebasis zu erstellen, endlich erfüllen.

## Malik

Ich habe das Thema gewählt, weil darin zwei meiner grössten Interessensgebiete vereinigt sind: Die Elektronik und die Neurowissenschaft. Vor allem bei letzterem kann ich hiermit mein Wissen erweitern und vertiefen. In der Elektronik kann ich vor allem mein bereits bestehendes Wissen anwenden und trainieren, was meinem späteren Berufsleben ein wenig zu gute kommen könnte.

## Danksagungen

Wir möchten unserem Abteilungsleiter, Oliver Schneider, in der MSW danken, welcher uns erlaubte an den Wochenenden in der MSW an unserer BMA zu arbeiten. Ohne seine Hilfe hätten wir unser Produkt sicher nicht rechtzeitig beenden können.

Daniel Büchler, Abteilungsleiter für die ersten beiden Lehrjahre der Elektroniker in der MSW hat uns freundlicherweise den Fräsbohrplotter zu Verfügung gestellt. Dieser Umstand hat unsere Arbeit ungeheuer beschleunigt.

# Abstract

# Einleitung

Das Ziel unserer BMA ist es, den Vorgang des Lernens mit greifbaren elektrischen Bauteilen zu simulieren. Dieser Aspekt allein hat schon eine gewisse Faszination an sich. Unter Machine-Learning versteht man etwas extrem Komplexes, was sich auf Softwareebene abspielt. Für eine Künstliche Intelligenz sind riesige Mengen an Daten Nötig um ihr etwas beizubringen. Aber wie so vieles, lässt sich auch eine Intelligenz, ob sie jetzt künstlich ist oder nicht, herunterbrechen auf ihre Elementarsten Teile. Beim Menschen sind es die Neuronen, bei der Maschine elektrische Schaltkreise. Unser Produkt ist ein wenig von beidem. Denn wir simulieren das menschliche Neuron mit einer elektrischen Schaltung.

Aufgrund unseres Lehrberufs, Elektroniker, setzten wir den Fokus auf die Funktionalität und nicht auf Biologische Richtigkeit. Das bedeutet, dass wir versuchten die chemischen Prozesse elektronisch zu imitieren. Es fielen zwar viele der Biologischen Aspekte eines Neurons weg, jedoch bleibt der Kern des ganzen schlussendlich bestehen: das Lernen.

Um die Printplatte herzustellen, auf welcher wir unsere Schaltung aufgebaut haben, verwendeten wir den MSW-Internen Fräsbohrplotter, welcher uns die ersten Prototypen in höchstgeschwindigkeit zur Verfügung stellen konnte.

Die Frage die wir uns mit dieser BMA beantworten wollen lautet:

Was ist eine Möglichkeit ein Neuron (Nervenzelle) mittels Elektronischen teilen zu realisieren und kann damit eine selbstlernende elektrische Schaltung (“Neuron”) erstellt werden?

Um diese Frage dreht sich die ganze Arbeit unserer BMA. Darum auch der Name Hardware-Neuron. Unser Endprodukt wird Ausschlisslich aus Elektronischer Hardware bestehen. Mit unserem Endprodukt werden wir uns den ersten Teil der Frage also selbst beantworten.

Zu Beginn des Hauptteils wird das Konzept zur Lösung des Problems in all ihren Details und mathematischen Formeln erklärt. Danach wird die Schaltung erklärt, die wir erstellt haben und nach den zuvor erläuterten Funktionen arbeitet. Darauf folgen das Design des Prints auf EAGLE und die Auflistung der Bauteile welche auf den Print gehören.

# Hauptteil

# Theorie

## Grundlage der Problematik

Es soll eine Schaltung erstellt werden, die Mathematische Berechnungen eines Neurons imitiert. Dadurch kann eine lernende Schaltung entwickelt werden, die dynamische bzw. lernende Schaltkreise ermöglichen. Zudem wird die Rechenzeit für ein Neuron reduziert.

Als lernen ist zu verstehen, dass die Schaltung berechnet, wie sie welche Eingänge verarbeiten soll. Das passiert, indem der Nutzer am Eingang Daten anlegt und gleichzeitig einen Sollwert für den Ausgang definiert. Mit der Zeit lernt also die Schaltung, wie sie die Eingangsdaten verarbeiten soll und wie sie darauf reagieren soll.

## Testbedingungen

Damit das Folgende auch gut verstanden werden kann, eine kurze Erklärung zu den Fachbegriffen. In der Elektronik bzw. Digitaltechnik wird mit dem Binären Zahlensystem gearbeitet, diese Zahlen werden aus Einsen und Nullen gebildet.

Jede 0/1 hat einen Zahlenwert den sie repräsentiert. Im binärsystem wird die Wertung von rechts nach links pro Stelle verdoppelt. D.h. die erste Ziffer von rechts steht für eine eins, die zweite für eine 2, die dritte für eine 4. Steht eine Eins an der Stelle wird die jeweilige Wertung dazu addiert, steht da eine Null, dann nicht.

Bsp.:

0001 entspricht im Dezimalsystem der Zahl 1. Weil: (0\*8)+(0\*4)+(0\*2)+(1\*1)=1

0010 entspricht der Zahl 2. Weil: (0\*8)+(0\*4)+(1\*2)+(0\*1)=2

0011 entspricht der Zahl 3. Weil: (0\*8)+(0\*4)+(1\*2)+(1\*1)=3

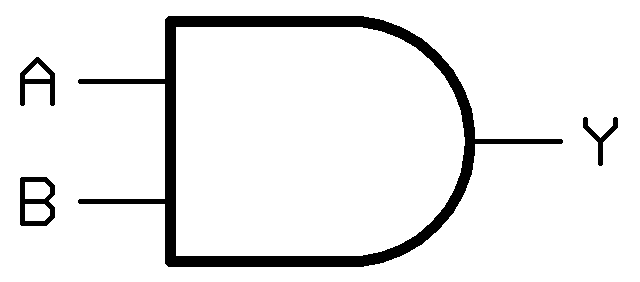
0100 entspricht der Zahl 4. Weil: (0\*8)+(1\*4)+(0\*2)+(0\*1)=4

Usw.

In der Praxis entspricht die 1 einer Spannung von 5 Volt (auch «High» genannt) und die 0 einer Spannung von 0V (auch «Low» genannt).

Um diese Informationen in der Elementarsten weise zu verarbeiten werden Drei Grundgatter verwendet:

AND:

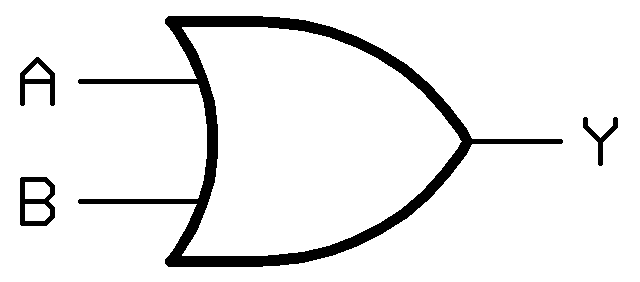


*(Quelle :* [*https://de.wikipedia.org/wiki/Datei:Logic-gate-and-us.png*](https://de.wikipedia.org/wiki/Datei:Logic-gate-and-us.png)*)*

|  |  |  |
| --- | --- | --- |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Das AND-Gate verknüpft die Eingänge (A & B) so, dass der Ausgang (Y) nur einem «High» entspricht, wenn die A und B zur gleichen zeit auch «High» sind. Jede andere Kombination führt zu einem «Low» am Ausgang.

OR:

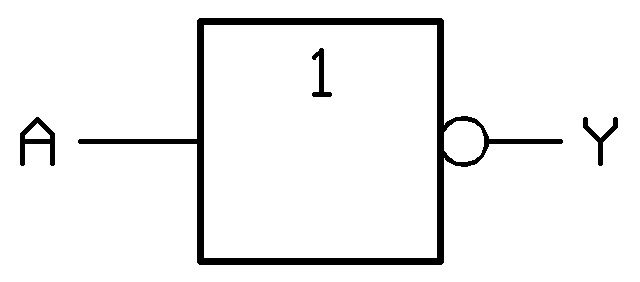


*(Quelle :* [*https://de.m.wikipedia.org/wiki/Datei:Logic-gate-or-us.png*](https://de.m.wikipedia.org/wiki/Datei:Logic-gate-or-us.png)*)*

|  |  |  |
| --- | --- | --- |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Das OR-Gate verknüpft die Eingänge (A & B) so, dass der Ausgang (Y) einem «High» entspricht, wenn mindestens einer der Beiden Eingänge «High» ist.

NOT:



*(Quelle :* [*https://de.wikipedia.org/wiki/Datei:Logic-gate-inv-iec.png*](https://de.wikipedia.org/wiki/Datei:Logic-gate-inv-iec.png)*)*

|  |  |
| --- | --- |
| A | Y |
| 0 | 1 |
| 1 | 0 |

Das NOT-Gate besitzt nur einen Eingang (A). Das Gatter macht nichts anderes, als das Gegenteil vom Eingang auszugeben.

### Möglichkeiten

Um zu überprüfen ob das Neuron auch lernt, soll es die Aufgabe von logischen Gattern (AND, OR) übernehmen bzw. lernen. Um zu überprüfen, ob das Neuron in der Lage ist zu merken, wenn ein Eingang nicht benötigt wird, wird neben den notwendigen Eingängen auch noch ein überflüssiger Eingang benutzt, der Ignoriert werden soll. Wenn dies zuverlässig funktioniert, kann man davon ausgehen, dass das Neuron in der Lage ist, bedingt komplexe Zusammenhänge zu approximieren und wichtige von nicht wichtiger Information zu unterscheiden.

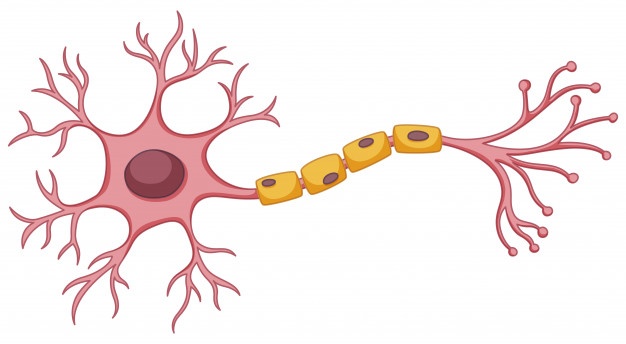
### Unmöglichkeiten

Da das Neuron auf keinen Fall in der Lage ist Invertierungen (NOT-Gatter) zu imitieren, sind Kombinationen auf der Basis: «Wenn ein Eingang **nicht** eingeschaltet ist, dann…». Von vorherein ausgeschlossen und unmöglich für das Neuron zu erlernen.

Um es in technischer Sprache zu sagen, das Neuron ist nicht in der Lage XOR, XNOR, NAND, NOR oder NOT Gatter zu erlernen, da nach dem Aufbau des Eingangs ([Inputs](#_Input)) immer ein Stromzufluss herrschen muss.

## Grundlegendes Model eines Neurons

### Biologisch



## 

Abbildung 1

Das Neuron ist eine Nervenzelle, die nahe zu überall in unserem Körper vorhanden ist. Ihr wichtigster Einsatzort ist das Gehirn.

## Bedingungen/Einschränkungen

Es gibt gewisse Probleme bei der Übersetzung vom Modell in die reale Welt. Zum einen ist die Praxis nie perfekt. Bauteile haben Toleranzen, Unendlichkeiten gibt es nicht und es ist nicht möglich Spannungen auf eine einfache Art und Weise zu multiplizieren.

Die gewünschte Formel wird auch nicht in einer reinen Form vorkommen.

## Typ des Neurons (Vorteile gegenüber anderen Methoden)

Als Grundprinzip benutzen wir das Modell eines Spiking-Perceptrons, was ein einzelnes Neuron mit einem pulsierenden, digitalen Ausgang darstellt.

Der Vorteil gegenüber anderen Methoden ist, es kommt der Biologie am nächsten und die Rückrechnung ist mit einzelnen Transistoren machbar.

Der Nachteil ist, es ist zeitabhängig. Was bedeutet, dass zeitlich alles gut aufeinander abgestimmt sein muss.

## Mathematischer Ansatz vorwärts Rechnung

Die Grundformel für den Kern des Neurons ist:

(Formel 1) Quelle: 7146.pdf

Wenn der Kern-Kondensator eine Gewisse Spannung erreicht wird ein «Spike» erzeugt, was am Ausgang ausgegeben wird. Dieser «Spike» wird durch eine Aktivierungsfunktion erzeugt.

## Aktivierung

: Ausgabe der Schaltung

Sobald der Spike erzeugt wird, wird über den Discharge-Transitor auf 0V gezogen. Was zur Folge hat, dass wenn der Input dauerhaft auf 5V gestellt ist, sich ein «Spike Train» mit einer Festen Frequenz bildet.

## Fehlerberechnung

Damit das Neuron überhaupt etwas lernt, muss eine Art Feedback vom Nutzer oder von der Umgebung kommen. Dafür wird ein Rückgabewert () verwendet. Mit diesem Rückgabewert kann die Gewichtung angepasst werden, indem verändert wird.

*(Bild 2) Quelle: 7146.pdf*

: Kondensator am Gate des Transistors

: Spannung am Eingang

: Widerstand der die Ent-/Lade Geschwindigkeit des Kondensators angibt (Lern- Schrittgeschwindigkeit)

: Ziel Spike-Train, dass das Neuron erreichen soll

Da und zeitbegrenzte Pulse sind, wird der Kondensator nur ein klein wenig entladen/geladen was den Transistor am jeweiligen Eingang etwas mehr sperrt oder leitet. Dadurch können alle Inputgewichte über die Zeit so eingestellt werden, dass dem Ausgang entspricht.

Dies ist natürlich nur bis zu einer gewissen Komplexität möglich. Vor allem, da die Eingangsströme nicht negativ sein können.

## Problematik von Theorie und Praxis

Die Grösste Problematik ist, dass das Speichern von Analogen Spannungen nur sehr schlecht geht und sehr störungsanfällig ist. Man kann die Schaltung etwas Stabilisieren, indem man den Print so gut wie möglich isoliert um jeglichen Energieverlust über benachbarte Leitungen zu vermeiden.

## Blockschaltbild

Output

Schaltung

Stecker

- 3 Eingänge

- Sollwert für Output

- Rückrechnung Ein/Aus

Abbildung 2

# Erklärung der Funktion

## Ganzes Neuron



Abbildung 3

Der Eingang (Input) verarbeitet die Spannung an den Eingängen bis und summiert sie. Das Zentrum (Core) des Neurons speichert die Werte des Eingangs über die Zeit, bis die Aktivierung (Activation) einsetzt und das Zentrum zurückgesetzt wird.

Der Eingang bei der Fehlerberechnung (DeltaCalcV2) entspricht dem Sollwert zu Trainingszeiten. Der Ausgang bei der Aktivierung wird zu jedem Zeitpunkt beim Training mit dem Sollwert mittels der Fehlerberechnung verarbeitet und der Eingang angepasst.

Das Neuron lernt somit über Zeit und der Eingang wird bei jedem Fehler um einen konstanten minimalen Schritt-Wert angepasst, um ein Schwingen um den Idealwert zu vermeiden und um das Neuron robuster gegen einzelne Fehler in den Trainingsdaten zu machen.

## Input



Abbildung 4

Die Eingangsspannungen bis werden über die, zum Teil leitenden, Transistoren in spezifische Ströme umgewandelt und am Ausgang summiert.

Die Spannungen an den Kondensatoren bestimmen, wie stark die Transistoren durchsteuern. So kann der Widerstand der Transistoren kontrolliert werden. Die Spannung an den Kondensatoren wird über die Fehlerberechnung verändert.

Im mathematischen Sinne entspricht diese Teilschaltung dem Abschnitt

Aus der Formel für die Vorwärtsrechnung.

Der kommt von dem Transistor , der direkt an angeschlossen ist.

### DeltaApply



Abbildung 5

Diese Teilschaltung ist für das Anpassen der Spannungen am Kondensator zuständig. Jeder Eingang besitzt eine solche Teilschaltung. Der Eingang Delta bestimmt, ob der Core-Kondensator geladen oder entladen wird. Enable schaltet mit der Spannung am Transistor den Rückrechnungsvorgang ein oder aus. Der Eingang sagt der Schaltung, ob der Eingang der Gesamtschaltung überhaupt aktiv war.

Diese Teilschaltung ist die Umsetzung des Formelabschnittes , wobei dem Enable-Eingang, dem Eingang und dem Delta Eingang entspricht.

## Core



Abbildung 6

Der Core besteht hauptsächlich aus einem Kondensator , welcher über den Eingang aufgeladen wird. Für den Fall, dass keine Signale über die Eingänge bis kommen, sorgt der Widerstand dafür, dass sich der Kondensator langsam entlädt und nach einer gewissen Zeit der Kondensator wieder zurückgesetzt ist.

Die Mathematische Darstellung der Teilschaltung ist . Dabei entspricht dem Ausgang der Input-Teilschaltung.

## Activation



Abbildung 7

Die Aktivierung ist dazu da, um zu bestimmen ob ein Signal weiter bzw. ausgegeben werden soll. Wenn die Spannung am Core (Kondensator) über die 2.5V steigt, schaltet der Vergleicher von 0V auf 5V über eine sehr kurze Zeit. Durch diese rapide Änderung gibt es eine Spannungsspitze auf der anderen Seite des Kondensators. Diese Spannungsspitze wird über die zwei folgenden Transistoren verstärkt, damit das Ausgangssignal eine höhere Spannung erreicht und auch mehr Strom liefern kann.

Dieses Ausgangssignal wird über die Grüne Leiterbahn zurückgeführt und entlädt den Core (Kondensator) sehr schnell. Wodurch dieser wieder auf unter die 2.5V fällt und der Ausgang des Vergleichers wieder auf 0V schaltet. Danach beginnt der Vorgang von vorne.

Die Diode am Ausgang sorgt dafür, dass wenn der Ausgang des Vergleichers auf 0V ist, der Ausgang der Ganzen Schaltung einen «Tri-state» annimmt (Das bedeutet, dass kein Strom rückwärts in die Schaltung fliessen kann).

### Vergleicher OPV



Abbildung 8

Dies ist eine primitive Schaltung eines Komparators bzw. Vergleicher, welcher in der Elektronik eine Essentielle Rolle spielt. Er vergleicht die Eingänge + & – und schaltet den Ausgang auf 5V, wenn die Spannung am + Eingang Grösser ist als die am – Eingang. Ist das umgekehrte der Fall wird der Ausgang auf 0V gesetzt.

## DeltaCalc



Abbildung 9

Die Delta-Calculation (DeltaCalc) ist dafür zuständig, zu entscheiden, ob der Ausgang (Ist-Wert) mit dem Soll-Wert übereinstimmt. Der Ausgang «Delta» von diesem Abschnitt gibt an, ob die Kondensatoren an den Eingängen bis geladen oder entladen werden sollen. Der Ausgang «En» (Kurz für Enable) von diesem Abschnitt sagt ob dieses Laden oder Entladen der Kondensatoren überhabt durchgeführt werden soll oder nicht.

In Digital-technischer Sprache:

Der Ausgang «Delta» entspricht einem Und Gatter mit den Eingängen und . Der Ausgang «En» ist eine XNOR Verknüpfung von und . Sprich wen dann sollen keine Änderungen an den Kondensatoren vorgenommen werden.

# Detailiertes Gesamtschema von EAGLE



Abbildung 10

# Messungen

## Blockschaltbild des Versuchsaufbaus

Mikro-kontroller

Schaltung

3Eingänge & Sollwert Sollwert

Istwert

Abbildung 11

Der Mikrokontroller ist so programmiert, dass in einem 10kHz Tackt hochgezählt wird und der momentan Wert, an den 3 Eingängen in binärer Form und den Sollwert mittels einer logischen Verknüpfung, an die Schaltung übertragen wird. Danach wird der Istwert der Schaltung vom Mikrokontroller eingelesen und abgespeichert. Pro Verknüpfung werden 100 Trainingsdurchläufe durchgeführt. Somit dauert ein Lerndurchgang ca. 70ms. Dies wird für die Verknüpfungen AND und OR durchgeführt und mehrmals nach einem Neustart der Schaltung wiederholt.

## Programmierung des Mikrokontrollers

# Benutzeranleitung

# Abkürzungsverzeichnis

* EAGLE einfach anwendbarer Grafischer Layout Editor

Wir haben dieses Programm verwendet, um darin unser Schema zu zeichnen und den Print zu designen

# Anhang

## Layout/Bestückungsplan von EAGLE



## Stückliste

