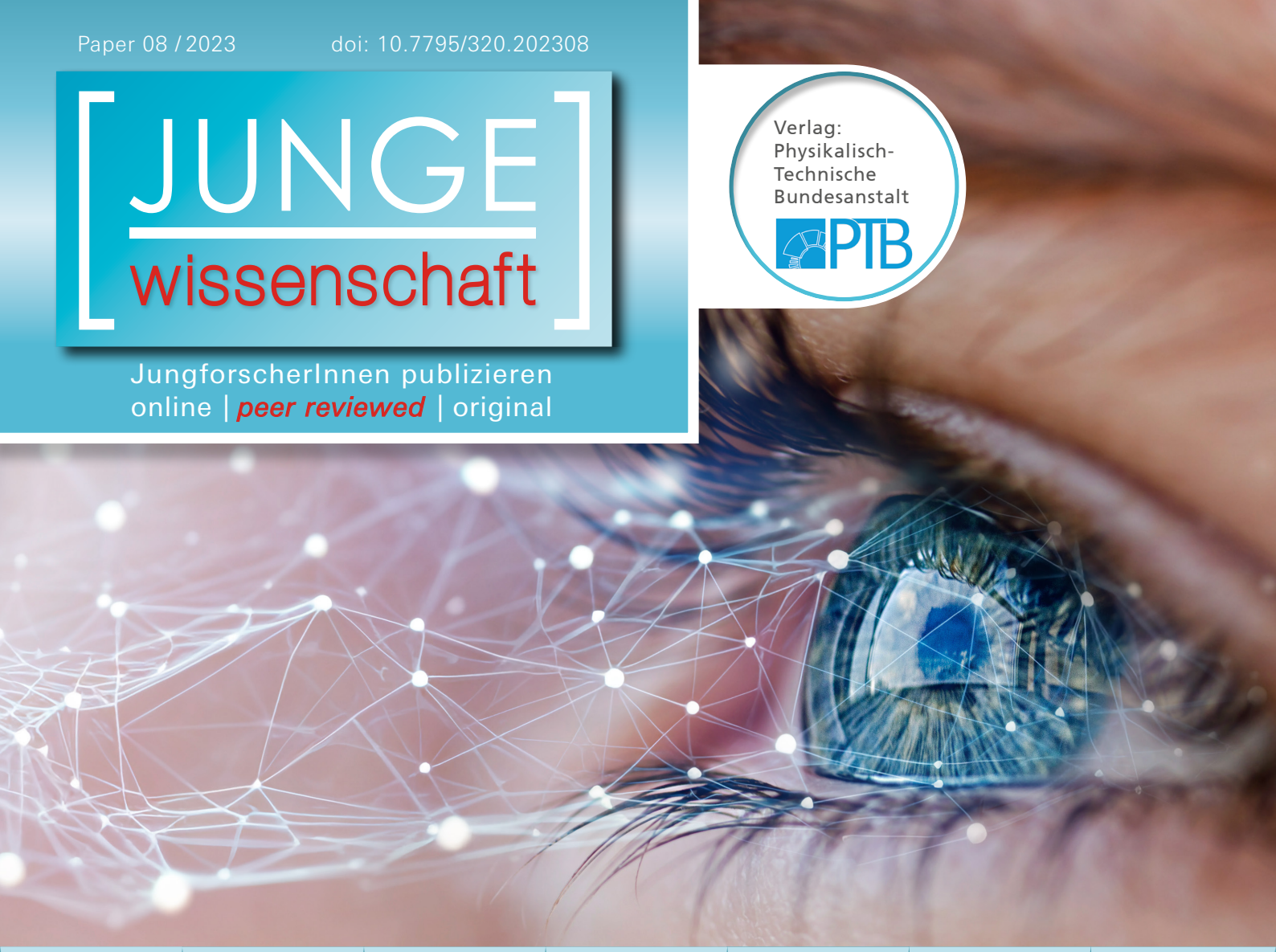
Paper 08 / 2023 doi: 10.7795/320.202308

JUNGE wissenschaft

JungforscherInnen publizieren
online | *peer reviewed* | original

Verlag:
Physikalisch-
Technische
Bundesanstalt
PTB



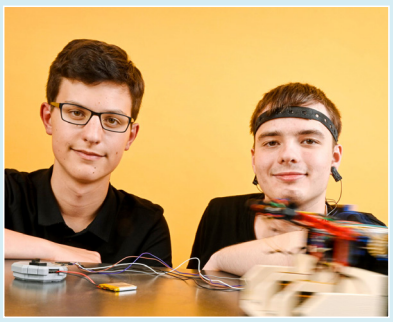
Mathematik & Informatik

Steuerung per Lidschluss

Erkennung ereigniskorrelierter Potentiale eines Elektroenzephalogramms durch ein neuronales Netz

Wenn man ein Augenlid schließt, erzeugt dies eine bestimmte elektrische Aktivität des Gehirns, die mit einem Elektroenzephalografen (EEG) gemessen werden kann. Die so erhaltenen Daten werden mithilfe eines neuronalen Netzes ausgewertet. So kann der Lidschluss anhand der EEG-Daten erkannt und genutzt werden, um einen Roboter fahren und anhalten zu lassen.

DIE JUNGFORSCHER



© Jugend forscht

Alexander Reimer (2006),
Matteo Friedrich (2007)
Gymnasium Eversten
Oldenburg

Eingang der Arbeit:
22.08.2022
Arbeit angenommen:
09.01.2023

Steuerung per Lidschluss

Erkennung ereigniskorrelierter Potentiale eines Elektroenzephalogramms durch ein neuronales Netz

1. Einleitung

Ziel des Projektes ist es, ein *Brain-Computer-Interface* (BCI) zu entwickeln, das erkennt, ob man gerade blinzelt. Dieses Erkennen soll dann zur Steuerung eines Roboters genutzt werden.

BCIs sind Schnittstellen zwischen dem Gehirn und einem Computer, die eine Kommunikation vom Gehirn zum Computer ermöglichen. Man kann BCIs für die Steuerung von Prothesen, Drohnen, Robotern und vielem mehr nutzen [1]. Zum Beispiel haben Forscher aus Stanford ein BCI entwickelt, welches in der Lage ist, behinderten Personen die Möglichkeit zu geben, über ihre Gedanken Nachrichten zu schreiben und so kommunizieren zu können [2]. Andere Forscher haben es geschafft, Menschen mit vollständigem Locked-in-Syndrom (CLIS) mithilfe eines BCIs wieder eine Kommunikation zu ermöglichen, trotz Fehlens jeglicher willentlicher Muskelbewegungen [3].

BCIs sind sehr teuer. Genau bei diesem Problem setzen wir mit unserem Projekt an: Es soll ein BCI entwickelt werden, das mit günstiger Hardware funktioniert. Dabei ist es wichtig, dass das BCI ohne spezielles Training auf eine bestimmte Person für jeden beliebigen Menschen funktioniert und leicht für die eigenen Zwecke anpassbar ist. Deswegen wird Open-Source-Software benutzt und der komplette Code ebenfalls auf GitHub veröffentlicht [18].

2. Grundlagen

2.1 Elektromyografie

Bei der Elektromyografie (EMG) werden Elektroden an der Hautoberfläche platziert. Diese können Spannungsdifferenzen messen, die durch Muskelbewegung entstehen [4]. Um eine Differenz bilden zu können, wird eine Referenzelektrode benötigt; in diesem Projekt wird sie am Ohrläppchen befestigt, wo sie ein von Muskelbewegung weitgehend unbeeinflusstes, konstantes Potential ableiten kann. Alle vier am Kopf platzierten Elektroden nutzen diese Referenzelektrode. Somit handelt es sich bei allen vieren um eine unipolare Ableitung [5].

2.2 Elektroenzephalografie

Die Elektroenzephalografie (EEG) ist im Verfahren identisch zur EMG, jedoch werden bei der EEG keine Muskelpotentiale gemessen, sondern sehr kleine Spannungen, die durch Potentialänderungen in Neuronengruppen im Gehirn entstehen und durch den Schädel dringen. Dies erreicht man, indem man die Elektroden an der Kopfoberfläche positioniert.

Eine Elektrode kann dabei meist nur die Summe aller lokalen Potentialänderungen messen. Man kann also auch nur ungefähr sagen, wo genau im Gehirn eine Potentialänderung stattgefunden hat. Mit mehr Elektroden kann die örtliche Genauigkeit erhöht werden [7].

2.3 Neuronales Netz

Ein neuronales Netzwerk besteht aus drei Teilen: der Eingabeschicht, den verdeckten Schichten und der Ausgabeschicht.

Die Eingabeschicht ist eine Liste aus Zahlen. Sie gibt an, welche Eingaben das Netzwerk bekommen soll, z. B. die EEG-Daten einer Testperson. Die verdeckten Schichten sind eine Ansammlung von in mehrere Schichten unterteilten Neuronen. Jedes Neuron besitzt eine Aktivierung, die als Zahl zwischen 0 und 1 angegeben werden kann, und einen *Bias* (Verzerrung), der eine beliebige Zahl ist. Die Neuronen verschiedener Schichten sind alle durch sogenannte Gewichte (*weights*) verbunden, die

doi: 10.7795/320.202308

ebenfalls einen beliebigen Wert haben. Die Ausgaben des neuronalen Netzwerkes sind dann die Aktivierungen der Neuronen in der Ausgabeschicht.

2.3.1 Forward Pass

Zur Berechnung der Aktivierung der Neuronen gibt es den sogenannten *Forward Pass*. Dabei beginnt man in der ersten verdeckten Schicht damit, für alle Neuronen die sogenannte Netzeingabe zu berechnen. Um die Netzeingabe eines Neurons zu berechnen, werden alle Aktivierungen der vorherigen Schicht mit den von dem Neuron dorthin führenden Gewichten multipliziert und aufsummiert. Der *Bias* ist eigentlich auch ein Gewicht, jedoch ist er mit einem Neuron verbunden, das immer die Aktivierung 1 hat.

Um aus dieser Netzeingabe nun die Aktivierung zu berechnen, benötigt man eine Aktivierungsfunktion, die dafür sorgt, dass die Aktivierung immer zwischen 0 und 1 liegt. In diesem Projekt wird dafür eine Sigmoidfunktion benutzt, die eine Zahl nimmt und einen Wert zwischen 0 und 1 ausgibt (siehe Abb. 1). Dies wird dann für jedes Neuron in jeder Schicht wiederholt. Da man aber immer die Aktivierungen der vorherigen Schicht benötigt, muss man das Ganze von der Eingabeschicht zur Ausgabeschicht durchführen. Daher auch der Name *Forward Pass*. Die Interpretation der Ausgaben hängt von den Trainingsdaten ab.

Die allgemeine Formel für die Aktivierung eines Neurons lautet also:

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

$$a_j = sig\left(\sum_L \left(a_L \cdot W_{Lj}\right) + b_j\right)$$

wobei a_j die Aktivierung des Neurons j,
L die vorherige Schicht,
a_L der Vektor aller Aktivierungen der Schicht L,
W_{Lj} der Vektor aller Gewichte zwischen dem Neuron j und den Neuronen der Schicht L, und b_j der *Bias* des Neurons j ist [14].

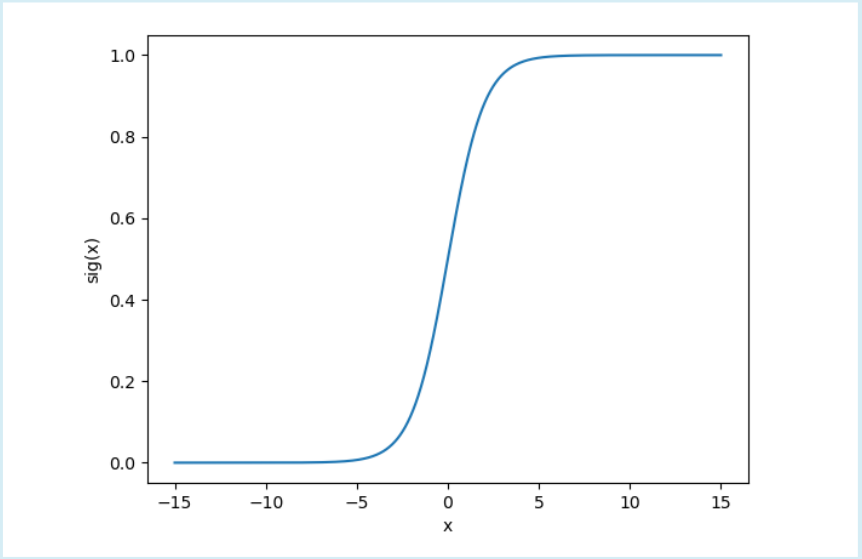


Abb. 1: Graph der Sigmoidfunktion. Die Sigmoidfunktion wird genutzt, um die Aktivierung *sig(x)* aus der Netzeingabe *x* zu berechnen.

Solche Schichten, in denen jedes Neuron der Schicht mit allen Neuronen der vorherigen und der darauffolgenden Schicht verbunden ist, nennt man *fully-connected* Schichten. Eine Beispielrechnung für ein winziges Netzwerk zeigt Abb. 2.

2.3.2 Kostenfunktion und Abweichung

Um zu bestimmen, wie gut ein neuronales Netz ist, gibt es die sogenannte Kostenfunktion (*cost function*), die die Datensätze verwendet, um die Abweichung des Netzwerkes vom Ideal zu bestimmen. Diese Abweichung wird auch *cost* genannt.

Trainingsdaten bestehen aus einer Liste von Trainingsdatensätzen. Jeder dieser Datensätze beinhaltet Eingaben für das Netzwerk und die richtigen Ausgaben dafür. Algorithmen des *Machine Learning*, die mit solchen Trainingsdaten arbeiten, werden überwachtes Lernen *(Supervised Learning*) genannt.

Die Funktion für den *cost* der Ausgabeschicht und somit des gesamten Netzwerkes (für einen Datensatz) lautet wie folgt:

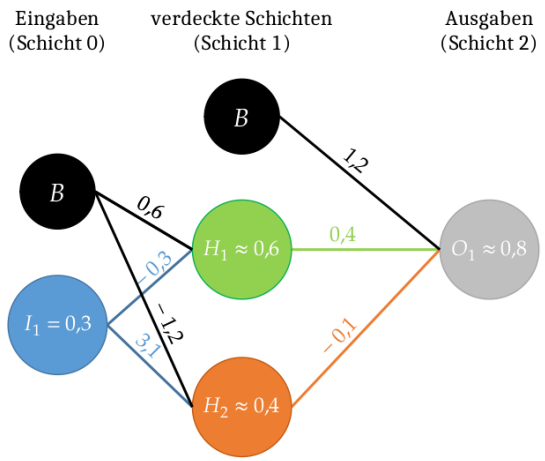
$$C_0 = (a_L - y)^2$$

wobei C_0 der *cost* der Ausgabeschicht, L die letzte Schicht (Ausgabeschicht), a_L der Vektor aller Aktivierungen der Schicht L, und y ein Vektor der richtigen Ausgaben für die Eingaben, mit denen die Aktivierungen berechnet wurden, ist.

Um den *cost* zu berechnen, muss man also für alle Neuronen der Ausgabeschicht die Differenz der durchs Netzwerk gegebenen und der in den Datensätzen vorgegebenen Aktivierungen bilden. Danach muss man diese Differenzen quadrieren und am Ende alle Ergebnisse aufsummieren. Dies kann man für alle Testdatensätze wiederholen und von allen *costs* den Durchschnitt nehmen, um die allgemeine Performance eines Netzwerkes zu überprüfen. Diese Art der Verlustfunktion wird mittlere quadratische Abweichung (*Mean Squared Error*, kurz MSE) genannt.

Zusammenfassend kann man also sagen, dass der *cost* die Abweichung von den aktuellen Ausgaben des Netzwerkes zu den idealen Ausgaben des Netzwerkes darstellt – ein perfektes neuronales Netz hätte einen *cost* von 0. Aufgrund der Trainingsdatensätze und der Verlustfunktion weiß man nun, wie stark die Ausgabeschicht abweicht und entsprechend verändert werden muss.

doi: 10.7795/320.202308



$$H_1(Netzeingabe) = 0{,}3 \cdot (-0{,}3) + 0{,}6 \approx 0{,}5$$

$$H_1(Aktivierung) \approx sig(0{,}5) \approx 0{,}6$$

$$H_2(Netzeingabe) = 0{,}3 \cdot 3{,}1 + (-1{,}2) \approx -0{,}3$$

$$H_2(Aktivierung) \approx sig(-0{,}3) \approx 0{,}4$$

$$O_1(Netzeingabe) \approx 0{,}4 \cdot 0{,}6 + (-0{,}1) \cdot 0{,}4 + 1{,}2 \approx 1{,}4$$

$$O_1(Aktivierung) \approx sig(1.4) \approx 0{,}8$$

Abb. 2: Ein einfaches Beispiel für ein neuronales Netzwerk mit 3 Schichten (Eingabeschicht, eine verdeckte Schicht und Ausgabeschicht), 4 Neuronen und 2 *Bias*-Neuronen. Die Neuronen sind durch Kreise mit der jeweiligen Aktivierung abgebildet. Die Linien zeigen die Verbindungen zwischen den Neuronen mit den dazugehörigen Gewichten. Die schwarzen Kreise zeigen die *Bias*-Neuronen, deren Aktivierung immer 1 ist.

2.3.3 Backpropagation

Durch die sogenannte *Backpropagation* (auch Fehlerrückführung) werden die Gewichte des Netzwerkes mithilfe der Datensätze angepasst, um den cost zu senken. Die von uns verwendete Form der *Backpropagation* ist das sogenannte Gradientenverfahren. Um die Aktivierungen der Ausgabeschicht anzupassen, müssen alle Gewichte und *Biases* davor angepasst werden. Um nun zu wissen, wie ein Gewicht verändert werden muss, gibt es beim Gradientenverfahren folgende Funktion:

$$\Delta W_{ij} = \varepsilon \cdot \delta_i \cdot a_j$$

Hier ist ΔW_{ij} der Wert, um den das Gewicht W zwischen den Neuronen j und i verändert werden muss, ε die Lernrate (meist ein kleiner Wert, wie 0,001), δ_i eine Annäherung der Ableitung des *cost* des Neurons i im Verhältnis zum Gewicht W_{ij}, und a_j die Aktivierung des Neurons j. Dabei ist oft verwirrend, dass die Schicht des Neurons i aus der Sicht des *Forward Pass* weiter hinten liegt als die Schicht des Neurons j.

Für den *Bias* wird die gleiche Formel benutzt, mit der Ausnahme, dass a_j immer 1 ist und somit wegfällt. Der Grund dafür liegt darin, dass der *Bias*, wie im Abschnitt 2.3.1 beschrieben, eigentlich nur ein Gewicht ist, das mit einem Neuron verbunden ist, welches immer eine Aktivierung von eins hat.

Um nun δ_i für die Ausgabeschicht zu berechnen, gibt es folgende Gleichung:

$$\delta_i = sig'(x_i) \cdot (a_i(\text{soll}) - a_i(\text{ist}))$$

wobei $sig'(x_i)$ die Ableitung von $sig(x_i)$ ist, also

$$sig'(x_i) = sig(x_i) \cdot (1 - sig(x_i)),$$

x_i die Netzeingabe des Neurons i, $a_i(\text{soll})$ die Aktivierung, die das Neuron haben sollte (also das gleiche wie y), und $a_i(\text{ist})$ die Aktivierung, die das Neuron hat.

Mit dieser Formel wird berechnet, welche Aktivierung das Neuron haben sollte, was an $(a_i(\text{soll}) - a_i(\text{ist}))$ erkennbar ist. Die Aktivierungsfunktion mit der Netzeingabe wird als Faktor mit einberechnet, da möglichst nur die Gewichte stark verändert werden sollen, die bei dem Trainingsdatensatz eine hohe Aktivierung haben, also durch diese Eingaben besonders angesprochen werden. So werden zum Beispiel beim Sortieren nur die Neuronen stark miteinander verknüpft, die für ein bestimmtes Muster verantwortlich sind.

Für die Neuronen der verdeckten Schichten muss man alle δ's der nächsten Schicht mit den von dem Neuron dorthin führenden Gewichten multiplizieren und dann summieren. Dadurch werden die Änderungen, die die Aktivierungen dieser Neuronen brauchen (δ), zusammengerechnet, da natürlich die Aktivierungen in der nächsten Schicht unterschiedliche Änderungen in dem gleichen Neuron benötigen.

Durch die Multiplikation mit den dahin führenden Gewichten werden diese Änderungen gewichtet, da sie auf einige Neuronen größere Auswirkungen haben als auf andere. Wie bei der Ausgabeschicht auch wird diese Summe noch mit $sig'(x)$ multipliziert, um die Aktivierung durch bestimmte Muster angesprochener Neuronen noch weiter zu erhöhen und die Aktivierung weniger bzw. kaum angesprochene Neuronen zu senken, sodass die Ergebnisse besser und eindeutiger werden. Die Formel hierfür lautet:

$$sig'(x_i) = \sum_L (\delta_L \cdot W_{Li})$$

wobei L die nächste Schicht, δ_L der Vektor aller δ's der Schicht L, und W_{Li} der Vektor aller Gewichte ist, die ein Neuron der nächsten Schicht und Neuron i verbinden.

doi: 10.7795/320.202308

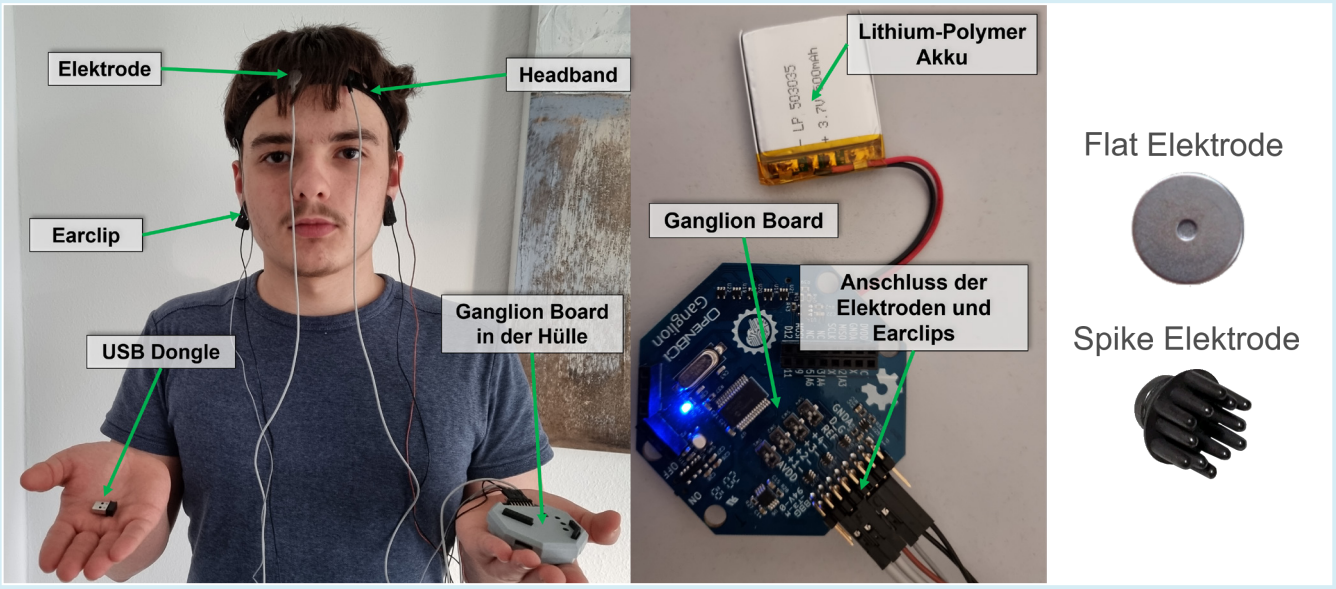


Abb. 3: Das EEG mit Zubehör. Links sieht man die typische Anwendung des EEG im Versuchsaufbau (Konfiguration 1, siehe Abb. 6). In der Mitte ist das Ganglion Board dargestellt. Rechts sind eine *Flat*-Elektrode und eine Spike-Elektrode zu sehen.

Da immer die (in Reihenfolge des *Forward Pass* gesehene) nächste Schicht benötigt wird, ergibt es Sinn, diese Optimierung bei der Ausgabeschicht zu starten und dann rückwärts die δ-Werte für jede Schicht zu berechnen und für die nächsten Berechnungen zu speichern – daher auch der Name *Backpropagation* [8, 15, 16].

3. Methode und Vorgehensweise

3.1 Materialien

Als Programmiersprache wurde Julia gewählt, da wir mit ihr durch vorherige Projekte gut vertraut sind. Durch die verfügbaren Softwarepakete sowie gute Performance ist sie für Datenverarbeitung, Mathematik und Statistik geeignet [19]. Für die Fourier-Transformation wurde das Softwarepaket FFTW.jl [20], für das neuronale Netz Flux.jl [23, 24] und zur Ausführung auf Nvidia GPUs CUDA.jl [25] verwendet.

Als EEG wurde das Ganglion Board von OpenBCI [9] gewählt (siehe Abb. 3, Mitte), da dieses einen niedrigen Preis hat (von uns angeschafft für ca. 550 €), quelloffen ist und die Daten mit BrainFlow.jl [21] leicht in Julia abgerufen werden können.

An die Platine können für unipolare Ableitungen bis zu vier Elektroden für Messdaten, eine Elektrode als Referenz und eine Elektrode als Erdung angeschlossen werden. Als Erdungs- und Referenzelektroden wurden zwei an jeweils einem Ohr befestigte Ohrclips verwendet (siehe Abb. 3, links). Wenn eine Elektrode an der Stirn befestigt wurde, wurde eine *Flat*-Elektrode verwendet, da diese eine größere Kontaktfläche hat und somit genauere Ergebnisse liefern kann. Bei einer Befestigung am Hinterkopf wurde eine Spike-Elektrode verwendet, da diese mit ihren Spitzen durch die Haare den Kontakt mit der Kopfhaut realisieren kann (siehe Abb. 3, rechts). Die Elektroden werden mit einem Klettband am Schädel befestigt, die Platine wird mit einer Plastikhülle geschützt (siehe Abb. 3, links).

Als Stromquelle wird für die Platine ein Lithium-Polymer-Akku verwendet (siehe Abb. 3, Mitte). Die Daten werden über einen *Bluetooth Dongle* an einen Laptop übertragen, sodass die Messeinheit mobil ist. Der Laptop, der zum Speichern und Verarbeiten der empfangenen Daten sowie für das Trainieren des neuronalen Netzes verwendet wird, hat einen 16 GB Arbeitsspeicher, eine RTX-3060 Grafikkarte und einen i7-11800H-Prozessor.

Für den Roboter werden LEGO-Technik- und Mindstorms-EV3-Teile verwendet, da unsere Schule diese bereits hat. Zum Fahren werden zwei große EV3-Motoren verwendet, welche an die Kontrolleinheit des Roboters, einen EV3-Brick, angeschlossen werden (siehe Abb. 4). Um den EV3-Brick und dadurch die angeschlossenen Motoren von einem externen Rechner in Julia steuern zu können, wird mit einer 8 GB SD-Karte das Betriebssystem ev3dev [10, 27] auf den EV3-Brick installiert und das selbst entwickelte Softwarepaket ev3dev.jl [22] verwendet. Mit einem USB-Kabel wird der EV3-Brick an einen Raspberry Pi 3B+ angeschlossen, der gewählt wurde, da er ein WLAN-Modul besitzt und zur Verfügung stand.

3.2 Vorgehensweise

Das Projekt lässt sich grob in drei Bereiche unterteilen. Zum einen gibt es den neurobiologischen Teil. Dieser be-

doi: 10.7795/320.202308

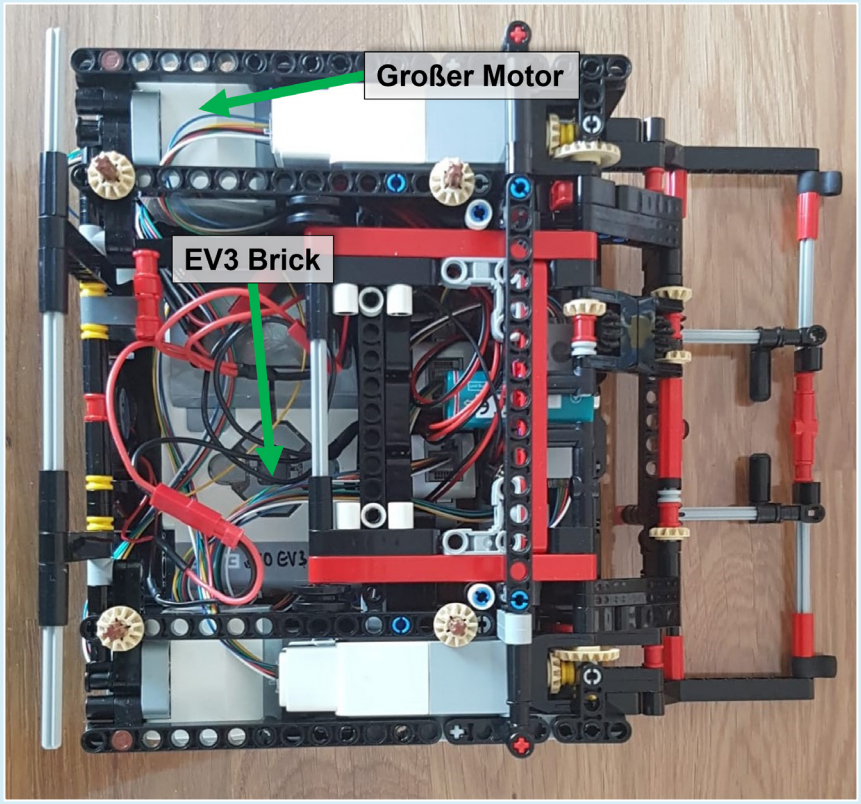


Abb. 4: Der verwendete Roboter (ohne *Raspberry Pi*, der über dem EV3-Brick angebracht wird). Die mittleren Motoren, die Konstruktion vorne, sowie die Sensoren vorne und unten sind für dieses Projekt nicht relevant.

steht aus der Messung der Gehirnaktivität und der Umwandlung dieser Aktivität in nutzbare Daten. Der zweite Teil besteht aus der Verarbeitung dieser Signale. Hierfür wird ein neuronales Netz, welches Muster in den Gehirnaktivitäten erkennen kann, benutzt. Der letzte Teil des Projektes beinhaltet die Steuerung eines EV3-Roboters. Je nachdem, was das neuronale Netz ausgibt, soll sich dieser Roboter anders verhalten und so über Gedanken steuerbar sein. Das heißt, dass der Roboter spezifische, zuvor definierte Aktionen wie zum Beispiel Vorwärtsfahren nach Belieben der Testperson ausführt.

3.3 Datenermittlung

Es wurde sich für die Erkennung von Blinzeln entschieden, da es im Graphen zumindest mit bloßen Augen sehr gut erkennbar ist (siehe Ausschlag nach 600 ms in Abb. 5).

Die Erkennung dieser EMG-Daten wurde als Vorversuch genutzt, da die Erkennung von derartigen Ausschlägen in EMG-Daten einfacher als die Erkennung von EEG-Daten ist. Außerdem sind EMG-Daten und EEG-Daten sehr ähnlich, was darauf schließen lässt, dass eine funktionierende Erkennung von EMG-Daten als Basis für eine Erkennung von EEG-Daten genutzt werden kann.

Zur Messung ereigniskorrelierter Potentiale (EKP) wurden zu Beginn zwei flache Elektroden (Elektroden mit glatter Kontaktoberfläche) über den Augen und zwei Spike-Elektroden (Elektroden mit langen Spitzen) an den Schläfen platziert (siehe Abb. 6).

Nach der Aufnahme der Trainingsdaten ist aufgefallen, dass bei den Elektroden an den Schläfen das Blinzeln deutlich schlechter erkennbar war als bei den Elektroden direkt über den Augen (siehe Abb. 5), vermutlich weil die Elektroden an den Schläfen Spike-Elektroden und weiter von den Augen entfernt waren. Deswegen wurden für die weitere Analyse nur die zwei Elektroden über den Augen verwendet.

Nachdem die Analyse der EMG-Daten funktioniert hat, wurde die Elektroenzephalografie genutzt. Dabei wurde weiterhin Blinzeln untersucht. Die Elektroden wurden dafür alle nebeneinander am Okzipitalen Kortex (Visueller Kortex) platziert (siehe Abb. 6 rechts), welcher sich am Hinterkopf befindet und sehr stark mit den Augen und dem Sehvermögen verknüpft ist. [6]

Es wurde entschieden, trotz des Umstiegs auf EEG bei Blinzeln als EKP zu bleiben, da geschlossene Augen in den EEG-Daten klar erkennbar sind: Bei offenen Augen findet eine sogenannte Alpha-Blockade statt, bei der – vor allem im Okzipitalem Kortex – die Alphawellen (ca. 7–13 Hertz) stark sinken. Entsprechend steigen die Alphawellen stark an, wenn die Augen geschlossen sind. Dieser Effekt wird auch Berger-Effekt genannt. [5, 11, 12]

Es wurde immer eine Sekunde EMG- oder EEG-Daten am Stück aufgenommen, sowohl für die Trainings- als auch für die Testdaten. Der Grund für diese Entscheidung ist, dass die Auswirkung von Blinzeln immer in diesem Rahmen erkennbar ist. Ein kürzeres Zeitintervall hätte im Prinzip auch funktioniert, jedoch ist die Fourier-Analyse (siehe unten) effektiver, je länger das Zeitintervall ist. Da jede Elektrode eine zeitliche Auflösung von 200 Hz (200 Messungen pro Sekunde) hat, werden also in der ersten Elektrodenkonfiguration (EMG) $2 \cdot 200 = 400$ und in der zweiten Elektrodenkonfiguration (EEG) $4 \cdot 200 = 800$ Signale pro Sekunde gemessen.

3.3.1 Fourier-Analyse

Die Fourier-Analyse wurde zur Vorbereitung der Daten für das neurona-

doi: 10.7795/320.202308

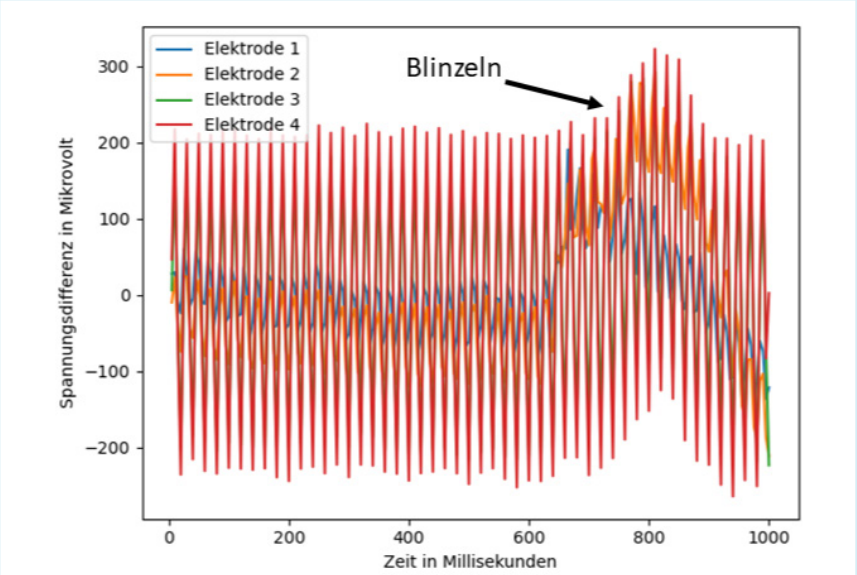


Abb. 5: Ausschnitt eines EMG beim Blinzeln (Konfiguration 1).

le Netz genutzt. Die Fourier-Analyse kann die verschiedenen zugrundeliegenden Frequenzen von Datenfolgen, Funktionen und mehr bestimmen, indem diese in Sinus-Kurven zerlegt werden, sie dient also zur Spektralanalyse. [17] In diesem Projekt wird dafür die Fast-Fourier-Transformation (FFT) genutzt, welche lediglich eine komplexere, aber effizientere Form der Diskreten-Fourier-Transformation (DFT) ist. [13]

Die Anwendung einer FFT zur Repräsentation von Gehirndaten ist sinnvoll, da verschiedene Funktionen und Aktivitätslevel von Gehirnbereichen durch die unterschiedlich getaktete und verzögerte Kommunikation zwischen einzelnen Neuronen gut durch Frequenzbänder repräsentiert werden können [vgl. 6, S. 469]. Demnach ist es auch logisch, dem neuronalen Netz die verschiedenen Frequenzen als Eingabe zu geben, da diese das Verhalten des Gehirns besser beschreiben als die rohen EEG-Daten.

Die FFT liefert eine Liste mit komplexen Zahlen. Der Index eines Wertes in der Liste bestimmt, für welche Frequenz der Wert gilt (erster Wert: 1 Hz, zweiter Wert: 2 Hz, etc.). Nun muss für jede komplexe Zahl der Abstand zum Ursprung bestimmt werden, also der absolute Betrag. Dieser entspricht dann der Amplitude der Frequenz. So lässt sich bestimmen, welche Frequenzen die größten Amplituden haben. Zudem können Frequenzen herausgefiltert werden, indem ihre Amplituden auf 0 gesetzt werden.

Bei 50 Hz kann eine große Amplitude entstehen, da die Wechselstromfrequenz in Deutschland 50 Hz beträgt. [5] Um dieses Störsignal herauszufiltern, wird bei der Verwendung von FFT die Amplitude für 50 Hz auf 0 gesetzt.

Vor der Implementierung der FFT wurde zuerst getestet, ob die so durchgeführte Spektralanalyse für unseren Zweck überhaupt sinnvoll ist. Dazu wurden die Ergebnisse der FFT von Elektroenzephalogrammen mit Blinzeln und ohne Blinzeln verglichen. Wie man in Abb. 7 sehen kann, gibt es aufgrund des Berger-Effekts vor allem im niedrigeren Frequenzbereich einen klaren Unterschied zwischen geöffneten und geschlossenen Augen. Das neuronale Netz sollte in der Lage sein, diesen Unterschied zu erkennen. Eine FFT ist also zur Vorverarbeitung der Daten geeignet.

Ein weiterer Vorteil der Verwendung der Spektralanalyse ist die Reduktion der Eingaben für das neuronale Netzwerk: Da der Betrag der komplexen Ergebnisse der FFT gebildet wird (nur die Amplituden sind für die Analyse notwendig), wird die Anzahl an Eingaben halbiert. Statt 200 Zahlen pro Elektrode pro Sekunde muss das neuronale Netzwerk also nur maximal 100 Zahlen pro Elektrode pro Sekunde verarbeiten. Außerdem ist, wie in Abb. 7 erkennbar, der Unterschied in den höheren Frequenzbereichen zwischen den beiden Zuständen vergleichsweise klein und vermutlich für eine sichere Erkennung nicht unbedingt notwendig. Somit könnten bei mangelnder Leistung des Netzwerks

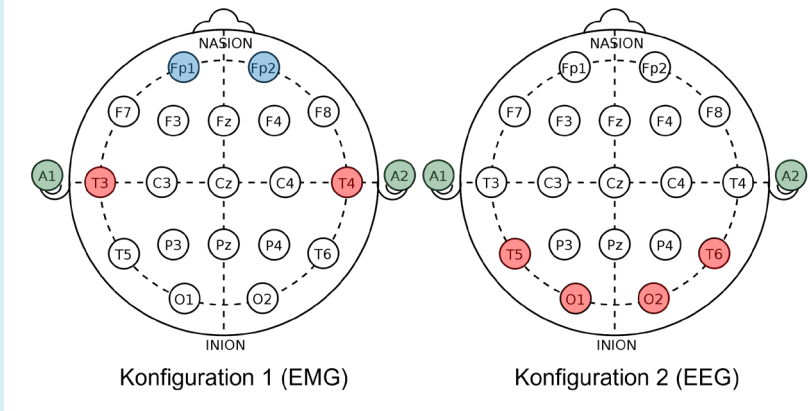


Abb. 6: 10-20 System mit den von uns genutzten flachen Elektroden blau, Spike-Elektroden rot, Erdung- und Referenzohrclip grün gefärbt. Links für EMG (Konfiguration 1), rechts für EEG (Konfiguration 2) [28].

doi: 10.7795/320.202308

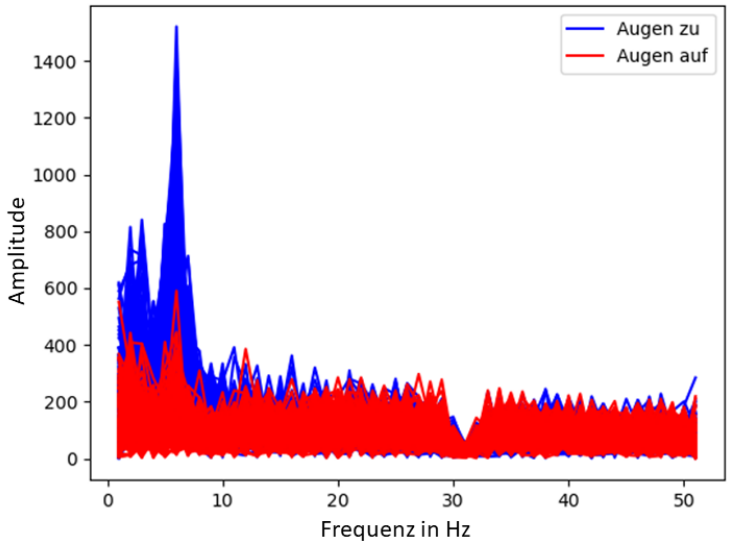


Abb. 7: Die Amplituden für die Frequenzen 1 Hz bis 50 Hz der Trainings- und Testdaten (EEG, Konfiguration 2), berechnet durch die FFT. blaue Linien: durchgehend geschlossene Augen, rote Linien: geöffnete Augen.

Amplituden für höhere Frequenzen als Eingaben ausgelassen werden.

Auf der anderen Seite gibt es praktisch keinen Nachteil der Spektralanalyse: Die FFT ist sehr schnell; auf unserem Test-System werden für die Verarbeitung von einer Sekunde Messdaten von zwei Elektroden im Schnitt 22 μs benötigt.

3.3.2 Der Roboter

Der Roboter besitzt LEGO-Mindstorms-EV3-Motoren, die von einem EV3-Brick gesteuert werden, der wiederum mit einem Raspberry Pi 3B+ verbunden ist und über ihn gesteuert werden kann (siehe Abb. 8 und 4). Der Grund dafür, dass die Motoren nicht direkt mit dem Raspberry Pi gesteuert werden, ist, dass schon alle Teile für einen EV3-Roboter vorhanden waren.

Auf dem EV3-Brick wurde mit einer SD-Karte das Betriebssystem ev3dev installiert. Dieses hat neben kompletem Zugriff auf das Debian-Betriebssystem des Bricks auch Zugriff auf die Motoren. Ev3dev übernimmt die direkte Kontrolle der Motoren und ermöglicht eine Steuerung bzw. einen Informationsabruf über verschiedene Dateien im /sys/class-Verzeichnis.

Auf dem Raspberry Pi wird das *Secure Shell File System* (kurz SSHFS) verwendet, um einen *Mount* zu erstellen, mithilfe dessen man über USB vom Raspberry Pi aus auf dieses /sys/class-Verzeichnis zugreifen kann.

Das Programm (Auswertung der EEG-Daten) sollte eigentlich direkt auf dem Raspberry Pi laufen. Dieser verfügt jedoch nur über einen Arbeitsspeicher von 1 GB. Dies stellt ein Problem dar, da nicht alle benötigten Softwarepakete geladen werden können. Deshalb muss ein Umweg gegangen werden: Anstatt das ganze Programm direkt auf dem Raspberry Pi am Roboter auszuführen, wird mit SSHFS auf unserem Laptop ein *Mountpoint* erstellt, welcher über WLAN auf den *Mountpoint* auf dem

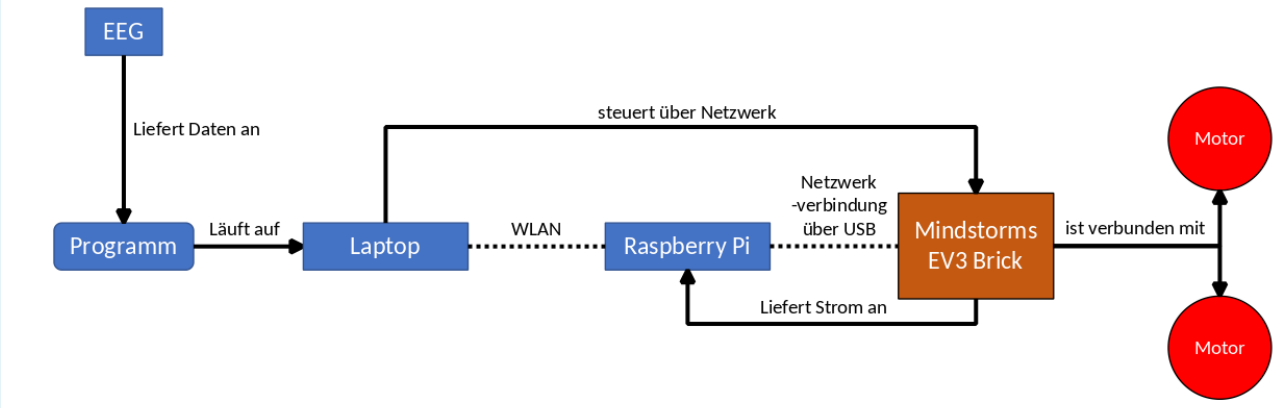


Abb. 8: Darstellung der Steuerungsabläufe in unserem Experiment. Die über das EEG aufgenommenen Daten werden auf dem Laptop durch das neuronale Netzwerk analysiert. Mithilfe des Raspberry Pis entsteht eine Netzwerkverbindung zwischen Laptop und EV3-Brick (gepunktete Linie). Der Laptop schaltet abhängig vom Ergebnis mittels des EV3-Bricks die Motoren an- oder aus.

doi: 10.7795/320.202308

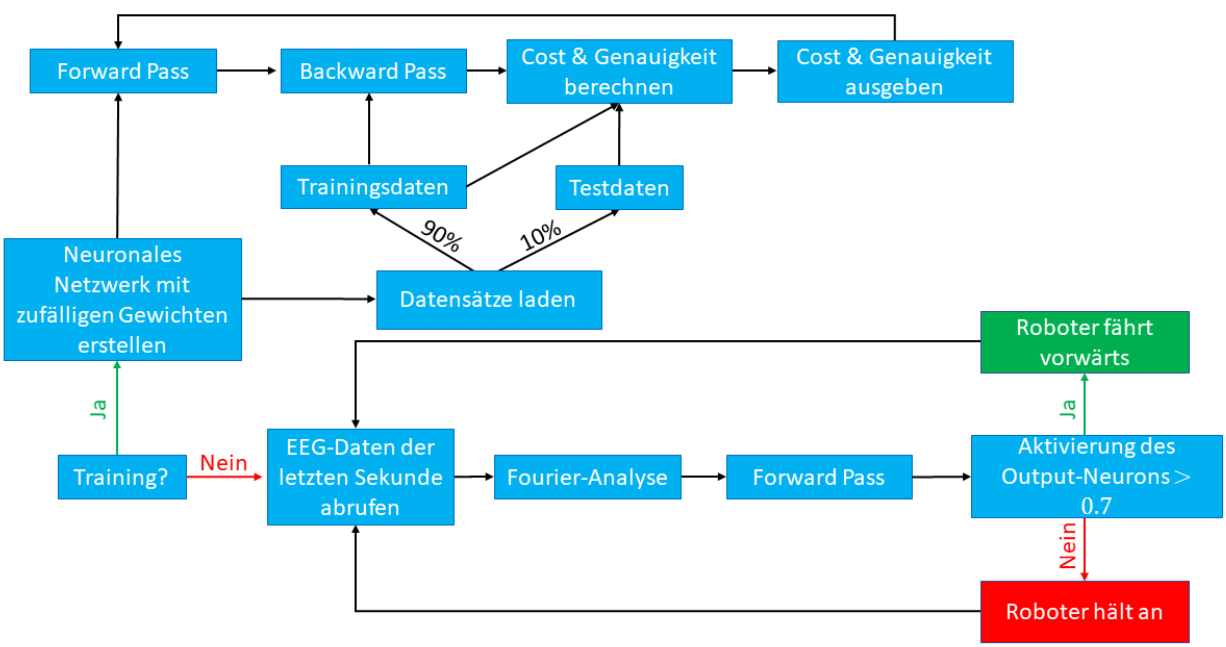


Abb. 9: Prinzipieller Programmablauf für Training des Neuronalen Netzes und Steuern des Roboters.

Raspberry Pi zugreift, und so über diesen Zugriff auf das /sys/class-Verzeichnis des Bricks hat (s. Abb. 8, gepunktete Linien). Das Auslesen der EEG-Daten, die Auswertung durch das neuronale Netzwerk und die Steuerung der Motoren können so auf dem Laptop stattfinden. Der Raspberry Pi wird also nur verwendet, um dem Laptop einen kabellosen Zugriff auf den Brick zu ermöglichen.

In einer von uns eigens entwickelten Julia-API wurden Funktionen implementiert, welche im Programm ein unkompliziertes Steuern des Roboters erlauben. Dabei machen die *low-level*-Funktionen nichts anderes, als z. B. die vom Programmierer gewünschte Geschwindigkeit in die entsprechende Datei zu schreiben. Der Code dazu wurde unter dem Namen ev3dev.jl auf GitHub veröffentlicht. [22]

3.3.3 Die Steuerung des Roboters mit den Augen

Der Roboter soll durch das Öffnen und Schließen der Augen steuerbar sein. Dies wird erreicht, indem der Roboter immer dann vorwärtsfährt, wenn die Aktivierung des Ausgabeneurons des neuronalen Netzwerks einen bestimmten Schwellwert überschreitet. Die Aktivierung des Ausgabeneurons gibt an, wie sicher sich das neuronale Netzwerk ist, dass die Augen zurzeit geschlossen sind. Der Roboter fährt also immer dann nach vorne, wenn die Augen geschlossen sind.

3.3.4 Übersicht über den Programmablauf

Beim Trainieren wird zuerst ein neuronales Netzwerk mit zufälligen Gewichten erstellt. Danach wird es mithilfe der Trainingsdaten, die 90 Prozent der vorher gesammelten Datensätze sind, trainiert. Mit den restlichen 10 Prozent, den Testdaten, sowie den Trainingsdaten, werden die aktuelle Genauigkeit und die *cost* berechnet. Dies wird wiederholt, bis eine zufriedenstellende Leistung erreicht wurde.

Wenn das Programm getestet wird, werden durchgehend die EEG-Daten der letzten Sekunde abgerufen, die Amplituden der Frequenzen mit FFT berechnet und diese an das neuronale Netz als Eingaben gegeben. Falls die Aktivierung des Neurons der Ausgabeschicht über dem Schwellenwert liegt (Augen geschlossen), dann fährt der Roboter vorwärts, falls die Aktivierung darunter liegt (Augen geöffnet), hält der Roboter an. Der Schwellenwert wurde durch Beobachtung in Tests sowie Ausprobieren bestimmt. In Abb. 9 ist der Ablauf als Diagramm dargestellt.

4. Ergebnisse

4.1 Analyse von EMG-Daten

In einem ersten Schritt wurden für die verdeckten Schichten des Netzwerks mehrere Strukturen ausprobiert. Letztlich wurde die in Abb. 10 gewählt, da sie konsistent gute Ergebnisse mit kurzen Trainingszeiten (ca. 3 Minuten für 1000 Epochen mit FFT) geliefert hat.

Für das Training des neuronalen Netzes wurden für die beiden Fälle (dass geblinzelt wurde bzw. dass nicht geblinzelt wurde) jeweils 100 Trainingsdatensätze aufgenommen. Die anschließend verwendeten Testdaten waren nicht in den Trainingsdaten enthalten, sondern sind zusätzlich aufgenommen worden.

doi: 10.7795/320.202308

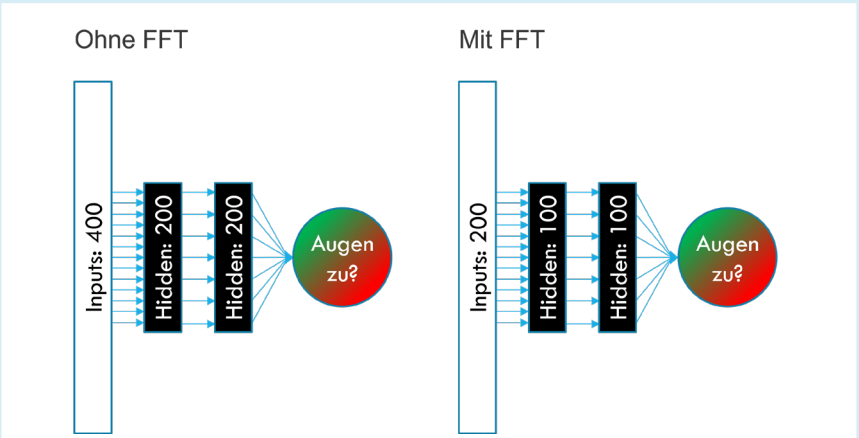


Abb. 10: Struktur des neuronalen Netzwerks bei Verwendung von EMG-Daten (Konfiguration 1), links ohne Vorverarbeitung mit FFT, rechts mit. Mit FFT gibt es pro Elektrode nur 100 Eingaben, da nur die Amplituden für Frequenzen 1 Hz bis 100 Hz verwendet werden.

Abb. 11 zeigt, dass eine Testdaten-Genauigkeit von 95 Prozent erreicht wurde, in einigen undokumentierten Durchläufen auch 100 Prozent. Mit Genauigkeit ist gemeint, welchen Anteil der gegebenen Datensätze das neuronale Netzwerk korrekt klassifizieren konnte.

4.2 Analyse von EEG-Daten

Für die Analyse der EEG-Daten wurde die Netzwerkstruktur in Abb. 12 gewählt.

Für das Training des neuronalen Netzes wurden jeweils 603 Datensätze für die Fälle, dass geblinzelt und nicht geblinzelt wurde, aufgenommen. 90 Prozent dieser Datensätze wurden zum Trainieren und 10 Prozent zum Testen des neuronalen Netzes verwendet.

Im Test konnte das neuronale Netzwerk ebenfalls eine Testdaten-Genauigkeit von 95 Prozent erreichen, jedoch nur mit vorheriger Spektralanalyse mittels FFT. Ein Live-Test ist im Video 1 zu sehen.

Ohne Spektralanalyse mittels FFT sah es deutlich schlechter aus (siehe Abb. 13): Die Genauigkeit mit den Trainingsdaten konnte zwar 100 Prozent erreichen, jedoch blieb die Testdaten-Genauigkeit, welche repräsentativer für die Leistung bei der Verwendung ist, bei 47,5 Prozent stehen. Der Grund dafür ist vermutlich ein Phänomen namens Überanpassung. Dabei passt sich das neuronale Netzwerk so stark an die Trainingsdatensätze an, dass leicht davon abweichende Daten nicht mehr erkannt werden. Dafür könnte ein zu langes Training verantwortlich sein oder, was in unserem Fall wahrscheinlicher ist, dass das neuronale Netzwerk ohne FFT keine tieferen Zusammenhänge finden konnte.

5. Diskussion und Ausblick

Das gesteckte Ziel wurde erreicht: Der Roboter fährt sicher, und es gibt kaum eine Verzögerung. Die Kosten der benutzten Hardware halten sich in Grenzen: Die gesamte EEG-Ausstattung hat ca. 550 € gekostet, was deutlich günstiger ist als professionelle Ausrüstung. Die Performance des Programms kann noch verbessert werden, jedoch dauern beim Trainieren 1000 Epochen (genug für ca. 100 Prozent Genauigkeit) auf dem Test-System lediglich fünf Minuten.

Die Anpassbarkeit ist begrenzt gegeben, wie unser Wechsel von EMG-Daten zu EEG-Daten zeigt. Als Nächstes wird unser Programm auch mit Personen

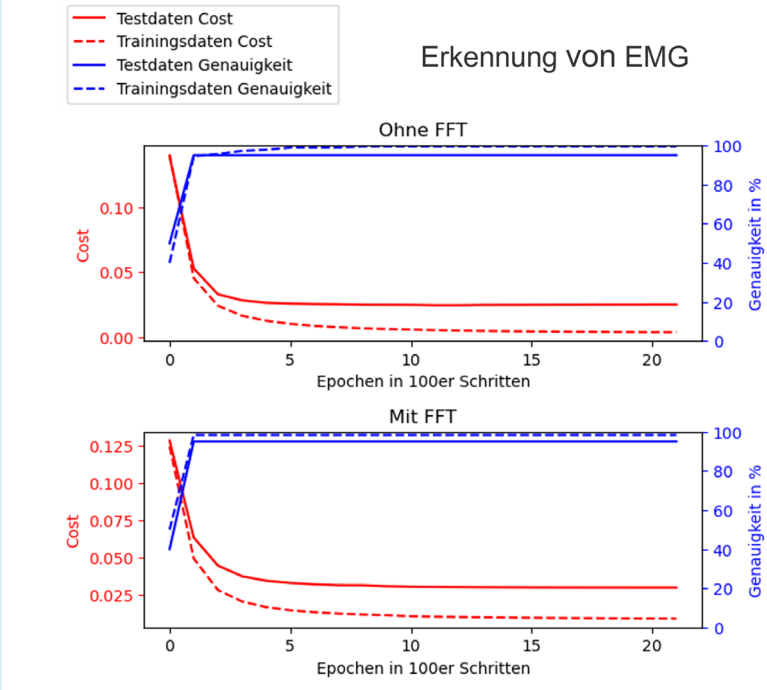


Abb. 11: Analyse der EMG-Daten mit (unten) und ohne (oben) Spektralanalyse mittels FFT. Eine Epoche entspricht jeweils einer *Backpropagation* mit allen Trainingsdatensätzen.

doi: 10.7795/320.202308

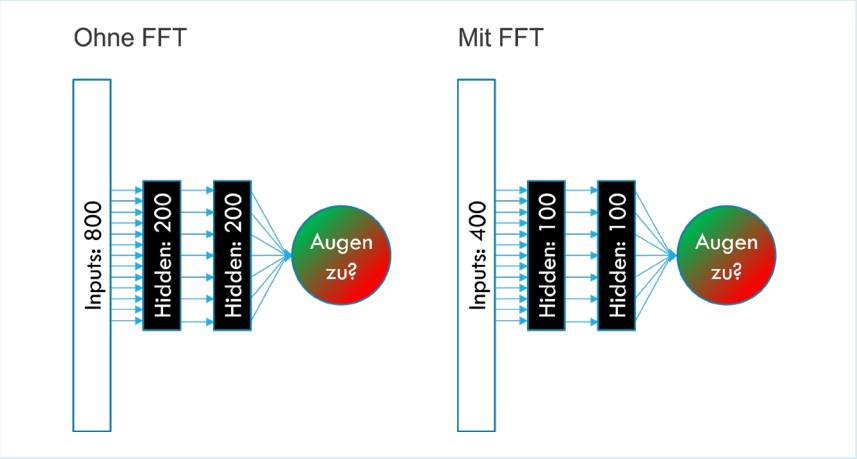


Abb. 12: Struktur des neuronalen Netzwerks bei Verwendung von EEG-Daten (Konfiguration 2), links ohne Vorverarbeitung mit FFT, rechts mit. Mit FFT gibt es pro Elektrode nur 100 Eingaben, da die Amplituden für Frequenzen 1 Hz bis100 Hz verwendet werden.

getestet, von denen vorher keine Trainingsdaten gesammelt wurden.

Ein Problem war die hohe Störanfälligkeit unserer EEG-Messausrüstung: Die Kabel durften nicht berührt werden und nicht zu viele technische Geräte in der Nähe sein. In einigen Räumen war die Signalqualität manchmal sehr stark gestört, ohne dass wir wussten warum. Also ist es zukünftig wichtig, mögliche Störfaktoren zu finden und zu eliminieren, damit konsistente Ergebnisse erreicht werden können.

Außerdem soll ein Raspberry Pi 4B mit 8 GB Arbeitsspeicher besorgt werden. Dieser sollte leistungsfähig genug sein, um unser Programm auch direkt auf dem Roboter ausführen zu können. So wären das EEG-Gerät und der Roboter auch ohne einen zusätzlichen Laptop nutzbar.

Eine weitere Verbesserungsmöglichkeit wird in der Fourier-Analyse sichtbar: Bei dem aktuellen Aufbau sind vor allem die niedrigen Frequenzen wichtig, da die Alpha-Blockade zur Erkennung genutzt wird. Die Bedeutung des niedrigen Frequenzbereichs kann man auch in Abb. 7 sehen. Es könnte sich also lohnen, z. B. nur die Amplituden der Frequenzen 1 Hz bis 20 Hz an das neuronale Netz als Eingaben zu geben, um die Leistung und Performance des neuronalen Netzes weiter zu steigern.

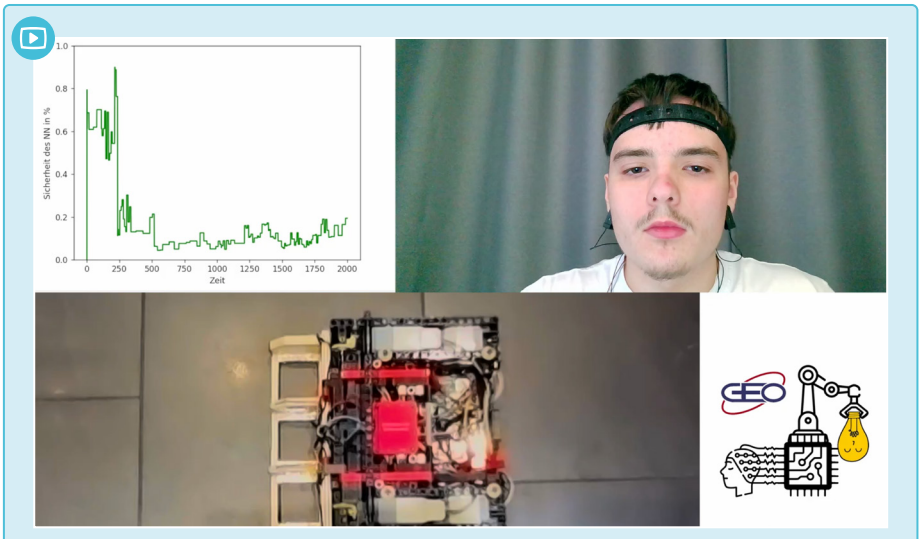
Eines der nächsten Ziele wird sein, auf andere ereigniskorrelierte Potentiale (EKP) wie Konzentration umzusteigen, und vielleicht sogar mehrere gleichzeitig zu erkennen, damit es für den Roboter mehr Kontrolldimensionen gibt, z. B. für Geschwindigkeit, rechts / links, etc. Man könnte für mehrere Befehle ein EKP nutzen, zum Beispiel zweimal schnell hintereinander Blinzeln für rechts, dreimal für links. Für eine Texteingabe könnte man versuchen, Protokolle wie Morsecode zu erkennen. Für eine korrekte Klassifizierung eines einzelnen Ereignisses (z. B. Augenschließen) braucht unser System eine Sekunde. Das heißt, dass Signale mit einer Datenübertragungsrate von 1 Bit pro Sekunde verarbeitet werden können.

Der Aufbau des verwendeten Netzwerks ist sehr einfach, da er nur aus den beschriebenen *fully-connected* Schichten besteht. Es gibt inzwischen schon fortgeschrittenere Arten von Schichten, wie z. B. *Convolutional Layer*, und Bauweisen, wie z. B. *Recurrent Neural Networks*. Zudem ist auch das Gradientenverfahren ein sehr grundlegender Optimierungs-Algorithmus. Inzwischen sind viel fortgeschrittenere Optimierungs-Algorithmen entwickelt worden, wie z. B. ADAM, was inzwischen praktisch der Standard im *Machine Learning* ist.

Danksagung

Danke an den Förderverein Gesellschaft der Freunde des Gymnasium Eversten e.V. für die Finanzierung des EEG-Geräts. Ohne diese Hilfe wäre dieses Projekt nie zustande gekommen.

Vielen Dank an Prof. Stefan Everling vom The Brain and Mind Institute der Western University in Kanada, der uns in einem Gespräch geholfen hat, einen



Video 1: Demonstration des Projektes. Immer wenn die Person die Augen schließt, fährt der Roboter vorwärts.

doi: 10.7795/320.202308

Ansatz für dieses komplizierte Thema zu finden, und praktische Tipps zum Umgang mit dem EEG gegeben hat, wie zum Beispiel die Anwendung einer FFT für die Vorverarbeitung der EEG-Daten.

Unser Dank geht auch an Ino Saathoff, der die Hülle für die EEG-Platine mit seinem 3D-Drucker erstellt hat. Wir danken ebenfalls Oliver Samkovskij, der die Lego-Struktur des Roboters gebaut hat, und Ino Saathoff, der für die Verkabelung und Stromversorgung verantwortlich war. Der Roboter ist zwar mit viel mehr ausgestattet als für unser Projekt notwendig, aber dank Euch musste kein neuer Roboter gebaut werden.

Unser größter Dank geht an unseren Projektbetreuer Dr. Ulf Glade, der unser Projekt begleitet hat. Er hat bei der wissenschaftlichen Methodik geholfen, uns wichtige Ressourcen gegeben sowie bei Fragen weitergeholfen.

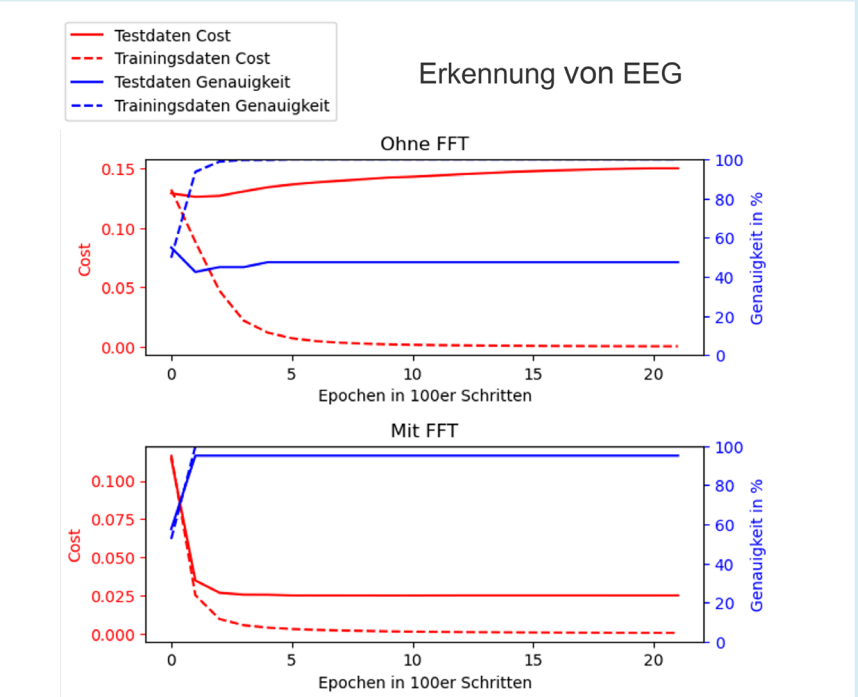


Abb. 13: Es wurde EEG benutzt. Zu sehen ist der *cost*- und Genauigkeitsverlauf der Test- und Trainingsdaten beim Trainieren des Netzwerks, oben ohne Vorverarbeitung durch FFT und unten mit. Eine Epoche entspricht jeweils einer *Backpropagation* mit allen Trainingsdatensätzen.

doi: 10.7795/320.202308

Literatur und Quellen

[1] Andrea Bonci u. a. „An Introductory Tutorial on Brain–Computer Interfaces and Their Applications“. In: *Electronics* 10.5 (Feb. 2021), S. 560. doi:10.3390/electronics10050560.

[2] Francis R. Willett u. a. „High-performance brain-to-text communication via imagined handwriting“. In: *bioRxiv* (2020). doi:10.1101/2020.07.01.183384. eprint: https://www.biorxiv.org/content/early/2020/07/02/2020.07.01.183384.full.pdf.

[3] Ujwal Chaudhary, Niels Birbaumer und Ander Ramos-Murguialday. „Brain-computer interfaces for communication and rehabilitation“. In: *Macmillan Publishers Limited* 12 (Sep. 2016), S. 513–525.

[4] Wikipedia. *Electromyography — Wikipedia, The Free Encyclopedia.* http://en.wikipedia.org/w/index.php?title=Electromyography&oldid=1070043736.2022. (Besucht am 16.02.2022).

[5] Universität Lübeck. *Elektroenzephalographie (EEG) und Ereigniskorrelierte Potentiale (EKP).* Praktikumsskript.

[6] N. Birbaumer und R. F. Schmidt. „Biologische Psychologie“. In: Springer Verlag, Berlin Heidelberg, 2010. Kap. 20.5, S. 468–493.

[7] Wikipedia. *Elektroenzephalografie – Wikipedia, The Free Encyclopedia*. http://de.wik ipedia.org/w/index.php?title=Elektroenzephalografie&oldid=216289194. 2022. (Besucht am 13.01.2022).

[8] Larry Hardesty. „Explained: Neural networks“. In: *MIT News* (2017). url: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

[9] OpenBCI. *Ganglion Specs*. url: https://docs.openbci.com/Ganglion/GanglionSpecs / (besucht am 01.05.2023).

[10] *Linux Kernel Drivers for ev3dev-stretch.* Dokumentation. url: https://docs.ev3dev.org/projects/lego-linux-drivers/en/ev3dev-stretch/.

[11] Helga Peter und Thomas Penzel. „Berger-Effekt“. In: *Springer Reference Medizin.* Springer Berlin Heidelberg, 2020, S. 1–1. doi: 10.1007/978-3-642-54672-3_350-1. (besucht am 30.03.2022).

[12] Wikipedia. *Berger-Effekt — Wikipedia, The Free Encyclopedia.* http://de.wikipedia.org/w/index.php?title=Berger-Effekt&oldid=208486396.2022. (Besucht am 29.03.2022).

[13] Sagar Khillar. „Difference Between FFT and DFT“. In: *Difference Between Similar Terms and Objects* (Sep. 2021). url: http://www.differencebetween.net/technology/difference-between-fft-and-dft/ (besucht am 18.02.2022).

[14] Brotcrunsher. *Neuronale Netze – Backpropagation – Forwardpass*. YouTube Video. 2017. url: https://www.youtube.com/watch?v=YIqYBxpv53A (besucht am 15.01.2022).

[15] Grant Sanderson 3blue1brown. *But what is a Neural Network?* YouTube Playlist. 2017. url: https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi (besucht am 10.01.2022).

[16] Brotcrunsher. *Neuronale Netze – Backpropagation – Backwardpass.* YouTube Video. 2017. url: https://www.youtube.com/watch?v=EAtQCut6Qno (besucht am 15.01.2022).

[17] Grant Sanderson 3blue1brown. *But what is the Fourier Transform? A visual introduction.* YouTube Video. 2018. url: https://www.youtube.com/watch?v=spUNpyF58BY (besucht am 10.01.2022).

[18] Alexander Reimer und Matteo Friedrich. *Interpreting EEG with AI*. GitHub Repository. 2022. url: https://github.com/AR102/Interpreting-EEG-with-AI.

[19] Jeff Bezanson u. a. „Julia: A Fresh Approach to Numerical Computing“. In: *SIAM Review* 59.1 (Jan. 2017), S. 65–98. doi: 10.1137/141000671.

[20] Matteo Frigo und Steven G. Johnson. „The Design and Implementation of FFTW3“. In: *Proceedings of the IEEE* 93.2 (2005). Special issue on “Program Generation, Optimization, and Platform Adaptation”, S. 216–231. doi: 10.1109/JPROC.2004.840301.

[21] Andrey Parfenov et al. *Brainflow.* GitHub Repository. 2018. url: https://github.com/brainflow-dev/brainflow.

[22] Alexander Reimer. *ev3dev.jl.* GitHub Repository. 2022. url: https://github.com/AR102/ev3dev.jl.

[23] Michael Innes u. a. „Fashionable Modelling with Flux“. In: *CoRR* abs/1811.01457 (2018). arXiv: 1811.01457. url: https://arxiv.org/abs/1811.01457.

[24] Mike Innes. „Flux: Elegant Machine Learning with Julia“. In: *Journal of Open Source Software* (2018). doi: 10.21105/joss.00602.

[25] Tim Besard, Christophe Foket und Bjorn De Sutter. „Effective Extensible Programming: Unleashing Julia on GPUs“. In: *IEEE Transactions on Parallel and Distributed Systems* (2018). issn: 1045-9219. doi: 10.1109/TPDS.2018.2872064. arXiv: 1712.03112 [cs.PL].

[26] Wikipedia. *Electrode locations of International 10–20 system for EEG (electroencephalography) recording.* https://en.wikipedia.org/wiki/10%E2%80%9320_system_(EEG)#/media/File:21_electrodes_of_International_10-20_system_for_EEG.svg. (Besucht am 08.07.2023).

Publiziere auch Du hier!

Forschungsarbeiten von Schüler/Inne/n und Student/Inn/en

In der Jungen Wissenschaft werden Forschungsarbeiten von SchülerInnen, die selbstständig, z. B. in einer Schule oder einem Schülerforschungszentrum, durchgeführt wurden, veröffentlicht. Die Arbeiten können auf Deutsch oder Englisch geschrieben sein.

Wer kann einreichen?

SchülerInnen, AbiturientInnen und Studierende ohne Abschluss, die nicht älter als 23 Jahre sind.

Was musst Du beim Einreichen beachten?

Lies die Richtlinien für Beiträge. Sie enthalten Hinweise, wie Deine Arbeit aufgebaut sein soll, wie lang sie sein darf, wie die Bilder einzureichen sind und welche weiteren Informationen wir benötigen. Solltest Du Fragen haben, dann wende Dich gern schon vor dem Einreichen an die Chefredakteurin Sabine Walter.

Lade die Erstveröffentlichungserklärung herunter, drucke und fülle sie aus und unterschreibe sie.

Dann sende Deine Arbeit und die Erstveröffentlichungserklärung per Post an:

Chefredaktion Junge Wissenschaft
Dr.-Ing. Sabine Walter
Paul-Ducros-Straße 7
30952 Ronnenberg
Tel: 05109 / 561508
Mail: sabine.walter@verlag-jungewissenschaft.de

Wie geht es nach dem Einreichen weiter?

Die Chefredakteurin sucht einen geeigneten Fachgutachter, der die inhaltliche Richtigkeit der eingereichten Arbeit überprüft und eine Empfehlung ausspricht, ob sie veröffentlicht werden kann (Peer-Review-Verfahren). Das Gutachten wird den Euch, den AutorInnen zugeschickt und Du erhältst gegebenenfalls die Möglichkeit, Hinweise des Fachgutachters einzuarbeiten.

Die Erfahrung zeigt, dass Arbeiten, die z. B. im Rahmen eines Wettbewerbs wie **Jugend forscht** die Endrunde erreicht haben, die besten Chancen haben, dieses Peer-Review-Verfahren zu bestehen.

Schließlich kommt die Arbeit in die Redaktion, wird für das Layout vorbereitet und als Open-Access-Beitrag veröffentlicht.

Was ist Dein Benefit?

Deine Forschungsarbeit ist nun in einer Gutachterzeitschrift (Peer-Review-Journal) veröffentlicht worden, d. h. Du kannst die Veröffentlichung in Deine wissenschaftliche Literaturliste aufnehmen. Deine Arbeit erhält als Open-Access-Veröffentlichung einen DOI (Data Object Identifier) und kann von entsprechenden Suchmaschinen (z. B. BASE) gefunden werden.

Die Junge Wissenschaft wird zusätzlich in wissenschaftlichen Datenbanken gelistet, d. h. Deine Arbeit kann von Experten gefunden und sogar zitiert werden. Die Junge Wissenschaft wird Dich durch den Gesamtprozess des Erstellens einer wissenschaftlichen Arbeit begleiten – als gute Vorbereitung auf das, was Du im Studium benötigst.

doi: 10.7795/320.202308

Richtlinien für Beiträge

Für die meisten Autor/Inn/en ist dies die erste wissenschaftliche Veröffentlichung. Die Einhaltung der folgenden Richtlinien hilft allen – den Autor/innen/en und dem Redaktionsteam

Die Junge Wissenschaft veröffentlicht Originalbeiträge junger AutorInnen bis zum Alter von 23 Jahren.

- Die Beiträge können auf Deutsch oder Englisch verfasst sein und sollten nicht länger als 15 Seiten mit je 35 Zeilen sein. Hierbei sind Bilder, Grafiken und Tabellen mitgezählt. Anhänge werden nicht veröffentlicht. Deckblatt und Inhaltsverzeichnis zählen nicht mit.

- Formulieren Sie eine eingängige Überschrift, um bei der Leserschaft Interesse für Ihre Arbeit zu wecken, sowie eine wissenschaftliche Überschrift.

- Formulieren Sie eine kurze, leicht verständliche Zusammenfassung (maximal 400 Zeichen).

- Die Beiträge sollen in der üblichen Form gegliedert sein, d. h. Einleitung, Erläuterungen zur Durchführung der Arbeit sowie evtl. Überwindung von Schwierigkeiten, Ergebnisse, Schlussfolgerungen, Diskussion, Liste der zitierten Literatur. In der Einleitung sollte die Idee zu der Arbeit beschrieben und die Aufgabenstellung definiert werden. Außerdem sollte sie eine kurze Darstellung schon bekannter, ähnlicher Lösungsversuche enthalten (Stand der Literatur). Am Schluss des Beitrages kann ein Dank an Förderer der Arbeit, z. B. Lehrer und Sponsoren, mit vollständigem Namen angefügt werden. Für die Leser kann ein Glossar mit den wichtigsten Fachausdrücken hilfreich sein.

- Bitte reichen Sie alle Bilder, Grafiken und Tabellen nummeriert und zusätzlich als eigene Dateien ein. Bitte geben Sie bei nicht selbst erstellten Bildern, Tabellen, Zeichnungen, Grafiken etc. die genauen und korrekten Quellenangaben an (siehe auch Erstveröffentlichungserklärung). Senden Sie Ihre Bilder als Originaldateien oder mit einer Auflösung von mindestens 300 dpi bei einer Größe von 10 · 15 cm! Bei Grafiken, die mit Excel erstellt wurden, reichen Sie bitte ebenfalls die Originaldatei mit ein.

- Vermeiden Sie aufwendige und lange Zahlentabellen.

- Formelzeichen nach DIN, ggf. IUPAC oder IUPAP verwenden. Gleichungen sind stets als Größengleichungen zu schreiben.

- Die Literaturliste steht am Ende der Arbeit. Alle Stellen erhalten eine Nummer und werden in eckigen Klammern zitiert (Beispiel: Wie in [12] dargestellt …). Fußnoten sieht das Layout nicht vor.

- Reichen Sie Ihren Beitrag sowohl in ausgedruckter Form als auch als PDF ein. Für die weitere Bearbeitung und die Umsetzung in das Layout der Jungen Wissenschaft ist ein Word-Dokument mit möglichst wenig Formatierung erforderlich. (Sollte dies Schwierigkeiten bereiten, setzen Sie sich bitte mit uns in Verbindung, damit wir gemeinsam eine Lösung finden können.)

- Senden Sie mit dem Beitrag die Erstveröffentlichungserklärung ein. Diese beinhaltet im Wesentlichen, dass der Beitrag von dem/der angegebenen AutorIn stammt, keine Rechte Dritter verletzt werden und noch nicht an anderer Stelle veröffentlicht wurde (außer im Zusammenhang mit **Jugend forscht** oder einem vergleichbaren Wettbewerb). Ebenfalls ist zu versichern, dass alle von Ihnen verwendeten Bilder, Tabellen, Zeichnungen, Grafiken etc. von Ihnen veröffentlicht werden dürfen, also keine Rechte Dritter durch die Verwendung und Veröffentlichung verletzt werden. Entsprechendes Formular ist von der Homepage www.junge-wissenschaft.ptb.de herunterzuladen, auszudrucken, auszufüllen und dem gedruckten Beitrag unterschrieben beizulegen.

- Schließlich sind die genauen Anschriften der AutorInnen mit Telefonnummer und E-Mail-Adresse sowie Geburtsdaten und Fotografien (Auflösung 300 dpi bei einer Bildgröße von mindestens 10 · 15 cm) erforderlich.

- Neulingen im Publizieren werden als Vorbilder andere Publikationen, z. B. hier in der Jungen Wissenschaft, empfohlen.

doi: 10.7795/320.202308

Impressum

Junge Wissenschaft
c/o Physikalisch-Technische
Bundesanstalt (PTB)
www.junge-wissenschaft.ptb.de

Redaktion
Dr. Sabine Walter, Chefredaktion
Junge Wissenschaft
Paul-Ducros-Str. 7
30952 Ronnenberg
E-Mail: sabine.walter@verlag-jungewissenschaft.de
Tel.: 05109 / 561 508

Verlag
Dr. Dr. Jens Simon,
Pressesprecher der PTB
Bundesallee 100
38116 Braunschweig
E-Mail: jens.simon@ptb.de
Tel.: 0531 / 592 3006
(Sekretariat der PTB-Pressestelle)

Design & Satz
Sebastian Baumeister
STILSICHER – Grafik & Werbung
E-Mail: baumeister@stilsicher.design
Tel.: 05142 / 98 77 89