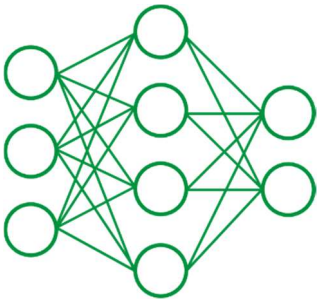
Ein neurales Netz zur Steuerung eines LEGO-Roboters

Experimente mit einer “KI” im Kleinstformat, um einen EV3 Roboter eine Linie folgen zu lassen.



Maturaarbeit
Gemäss den kantonalen Vorgaben des Kantons Graubünden
und den GKD-spezifischen Ausführungen

vorgelegt von

Gianpaolo Rossi

aus

Poschiavo / Schweiz

am Gymnasium Kloster Disentis

Genehmigt auf Antrag des Betreuers

Martin Huber

Und des Co-Referenten

Mischa Bachmann

Prada/ 20.10.2023

Inhaltsübersicht

1 Einleitung 1
2 Neuronale Netzwerke 2
2.1 Neuronen 2
2.2 Die Aktivierungsfunktion 3
3 Das Training 4
3.1 Die Kostenfunktion 5
4 Gradientenverfahren 5
4.1 Beispiel 7
5 Das Line Following mit LEGO EV3 Roboter 9
5.1 Der Roboter 9
5.2 Die Installation des Betriebssystems 10
5.3 Lösungsansatz mittels klassischer Regeltechnik 11
5.4 Ist das eine Aufgabe ein neuronales Netzwerk? 13
6 Ansatz der Lösung des Problems «Linefollowing» mit einem neuronalen Netz. 14
7 Funktionen und Ableitungen 14
8 Programmcode auf dem Roboter 17
8.1 Neuronales Netzwerk Für LEGO-Roboter 17
8.2 Ausführung des Programmes 18
9 Training des NN auf eigenen Rechner 19
9.1 Neuronales Netz 19
9.2 Kostenfunktion 20
9.3 Gradienten der Kostenfunktion 20
9.4 Update der Weights und Biases 21
9.5 Trainingsschritt mit Kontrolle von ε 21
10 Das Resultat 22
10.1 Versuch der Parameterbestimmung eines NN zur Validierung des Modelles 22
11 Diskussion/ Ausblick 25
12 Abstract 26
13 Literaturverzeichnis 27
14 Anhang 28
15 Eigenständigkeitserklärung 29

Vorwort des Autors

Seit ich mich erinnern kann, habe ich mich am Bau und der Funktionsweise von Geräten interessiert. Ich habe mich immer gefragt: «Wie funktioniert es? Und wie ist es gebaut?» Die Fragen führten zu immer neuen Themen: Erst zur Mechanik, dann in die Elektrotechnik, später allgemein in die Physik und Mathematik und zuletzt vermehrt zur Computertechnik mit ihren Sprachen, Betriebssystemen und Softwares, der Geschichte und den Geräten, die miteinander in einem weltweiten Netz verbunden sind. In der heutigen Zeit denke ich, dass es wichtig ist, sich über Computer-Wissenschaft zu informieren Zum, Beispiel über News von den grossen Techfirmen, welche über schnellere Computerchips oder sicherere Betriebssysteme berichten. In den letzten fünf Jahren haben wir die neue Entwicklung, betreffend Künstliche Intelligenz (KI) mitbekommen. Speziell im November 2022 hat es mit der Veröffentlichung von ChatGPT einen erneuten Boom. Denn jetzt hat jeder Zugriff zu grossen und starken KI-Modellen, welche das tägliche Leben erleichtern und effizienter machen können.
In dieser Arbeit habe ich versucht, grundlegende Informationen zu sammeln, welche die Basis im Bereich KI und im Besonderen von Machine Learning (ML) mit neuronalen Netzen sind. Das primäre Ziel war meine Kenntnisse in diesem Bereich zu erweitern und im Stande zu sein, die technischen Begriffe und Prozesse zu erklären, aber auch selbst in Python praktisch zu programmieren.

Während der Bearbeitung dieser Arbeit habe ich viele Informationen und Erlebnisse gesammelt, welche mich in der zukünftige akademische arbeite helfen werden. Mir wurde klar, dass Zeit- und Ressourcen-management verbesserungspotenzial aufweisen. An dieser Stelle danke ich: Herr Martin Huber für die ausführlichen Erklärungen und die grossartige Betreuung und Unterstützung. Ezio Rossi und Nathalie Pfiffner für die Sprachliche Unterstützung trotz der Beschränkte Zeit. Arno Rossi für die allgemeine Unterstützung. Marco Rossatti für die Hilfe bei Ausdrucken und Binden des Endresultates.

Prada, den 20.10.2023
Gianpaolo Rossi

1 Einleitung

Die Menschheit versucht immer wieder ihre Tätigkeiten zu erleichtern und zu automatisieren (BBC, 2023). zunächst durch die Nutzung von Tieren, welche einen Grossteil der schweren Arbeit übernahmen, danach durch technische Entwicklungen wie das Rad, das den Transport von schwereren Gegenständen ermöglichte. Später führte die Mechanisierung in Fabriken zu Massenproduktion von Waren und zu tieferen Kosten. Fabriken waren anfangs an Flüsse gebunden, aber nach der Erfindung von Dampfmotoren, Verbrennungsmotoren und elektrische Motoren war es überall möglich Fabriken zu bauen. Computer sind ein weiterer Schritt in Richtung Automatisierung. Diese komplexen Maschinen sind für gewisse Aufgaben schneller und präziser als jeder Mensch, sodass sie Probleme lösen können, welche für uns zu komplex und zu aufwändig sind.

Seit der Veröffentlichung von ChatGPT Ende 2022, hat die Menschheit ein AI-Hype entwickelt. Mittlerweile sind auch andere KI-Tools wie Dall-E, Bing-Chat, Google Bart und viele andere für die Öffentlichkeit jederzeit erreichbar. In Tech-Foren und Tech-Communities wird das Jahr 2023 schon als Jahr der KI genannt (Nilekani, 2023), da die grössten Techfirmen riesiges Kapital in viele KI-Startups investieren, wie beispielsweise Microsoft, welche 10 Milliarden Dollar an Open AI (Entwickler von ChatGPT) gaben (Bloomberg, 2023). Oder die vielen Investitionen, welche Firmen in ihre eigenen KI-Abteilungen gemacht haben, um den Wettlauf um die AGI (eng: Artificial General Intelligence) anzutreten (The New York Times, 2023). So scheinen diese Technologien immer mehr ein Teil unseres Lebens zu werden und darum ist es wichtig, dass wir verstehen, um was es dabei geht. Wie funktionieren diese verwunderlichen vieles könnenden Technologien? Wie sind sie gebaut?

Um ein Verständnis über eine grundlegende KI-Technologie, die NN zu erhalten, betrachten wir in dieser Arbeit das Problem im Kleinformat. Dafür hat es sich einen Challenge gestellt: Einen LEGO EV3 Roboter so zu programmieren, respektiv so einzulernen, dass er eine Linie auf dem Boden folgen kann. Bereits im Vorfeld war es klar, dass obwohl dies einfach scheint, eine echte Herausforderung ist.

Nach einer Einführung in den Grundlagen der KI, es wird ausführlich berichtet, wie diese spannende Aufgabe gelöst wurde. Dabei wurden die Grundlagen vertieft und mit dem konkreten Bezug auf die Aufgabe aufgearbeitet

2 Neuronale Netzwerke

Neuronale Netzwerke sind heutzutage ein fester Bestandteil von KI-Systeme. Diese werden mittels Machine Learning (ML) automatisch trainiert. Somit muss man nicht mehr jede einzelne Bedingung und das Verhalten einer Maschine selbst programmieren, sondern man muss, salopp gesagt, nur noch genug Daten, Computer, Energie und Zeit haben, um die meisten Probleme automatisch lösen zu können.

In diesem Kapitel geht es um das Fachwissen eines NN. Die Hauptquelle für dieses Kapitel ist das erste Kapitel von Neural Networks and Deep Learning (Nielsen, 2015) und das zweite Kapitel von Python deep Learning (Ivan Vasilev, 2019).

2.1 Neuronen

Das Neuron ist die kleinste Einheit einem neuronalen Netzwerk. Es ist der Ort, wo die wirklichen Berechnungen stattfinden.

$$y = f\left(\sum_i x_i w_i + b\right)$$

Ist die Formel für den Output eines Neurons, wo y der Output ist, x_i die Inputs, die üblicherweise von anderen Neuronen Stammen und w_i die Weights (gibt an, wie wichtig ein Input ist), b der Argument Bias, welche die Funktion durch zusätzlichen Freiheitsgrad berschiebt, und f ist eine «beliebige» Funktion. Für f wird oft die Sigma Funktion der Form $f(x) = \frac{1}{1+e^{-x}}$ benutzt eine andere oft benutzte Funktion ist die ReLu (en: rectified linear Unit) der Form $f(x) = \frac{x+|x|}{2}$ welche aber das Problem hat, dass bei x=0 nicht ableitbar ist.

Der eigentliche Zweck eines Neurons ist, sich von verschiedenem Werten zu aktivieren und eine grosse Zahl auszugeben oder nicht aktivieren und eine kleine Zahl auszugeben. Diese Sortierung passiert indem, verschiedenen Werte verschiedene Gewichte und Biases haben und werden dann an andere Werte und skalaren summiert bis dann das Ganze von einer Aktivierungsfunktion bereinigt wird. Wenn die Aktivierungsfunktion die Identitätsfunktion ist, wenn dem Neuron ein Negatives Zahl ausgibt, dann dem Neuron ist nicht aktiviert sonst, wenn der Neuron aktiviert wurde, dem Neuron gibt ein positiver Wert aus. Durch die Manipulation der Weights W und Biases B werden einige Werte wichtiger und andere sekundär.

2.2 Die Aktivierungsfunktion

Die Aktivierungsfunktion erlaubt dem Neuron die Ausgangswerte in verschiedene Interwalle aufzuteilen. Dies ermöglicht, dass die Output-Werte der Neuronen begrenzt werden können. Diese Eigenschaft der Neuronen ist nötig damit das neuronale Netzwerk auch lineare unteilbare Mengen teilen kann, in dem die Aktivierungsfunktion nicht mehr linear ist, sondern quadratisch kubisch, exponentiell oder sigmoidal ist. Einige Beispiele für Aktivierungsfunktionen sind die Folgende:

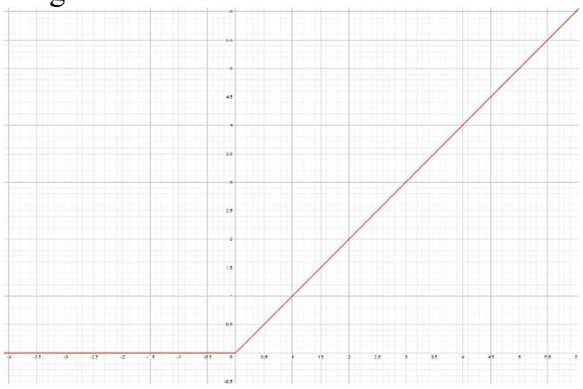


Abbildung 1 ReLu

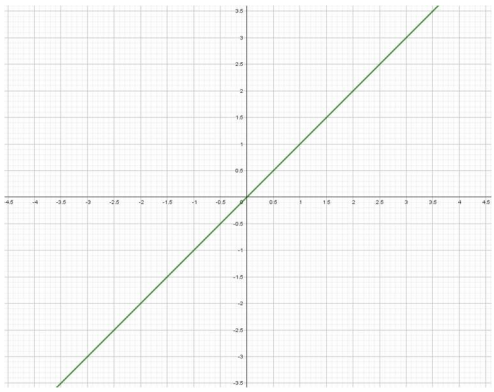


Abbildung 2 Identitätsfunktion

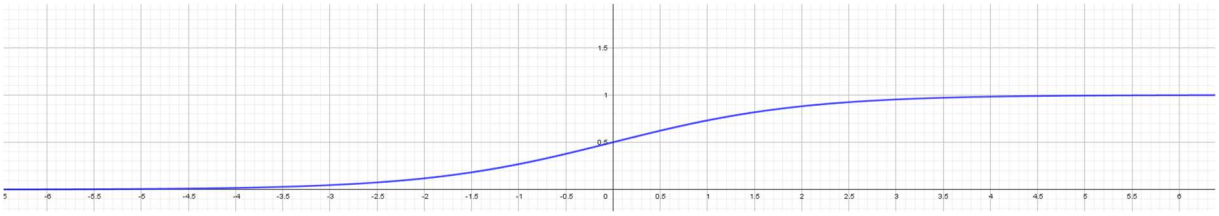


Abbildung 3 Sigma Funktion

Dem Neuron gibt Werte in verschiedene Mengen aus. wenn verschiedene Aktivierungsfunktionen benutzt werden, folgen verschiedene Wertemenge. Die Wertemenge für die Identitätsfunktion zum Beispiel $(f(x) = x)$ ist $\mathbb{W} = \mathbb{R}[-\infty, \infty]$, anders ist die Wertemenge einer Sigma-Funktion: $\mathbb{W} = \mathbb{R}[0,1]$. Das bewirkt im Output der Neuronen welche im Ersten Fall ein Beliebiges Wert Im $\mathbb{R}$ ausgibt, im zweiten Fall, nur zahlen zwischen 0 und 1 ausgegeben werden. Diese Variabilität in der Mengen bewirkt im Neuronales Netzwerk ein Vorteil. Wenn der Intervall beschränkt ist, dann wird der Anzahl der Werte innerhalb des neuronalen Netzwerks nicht ins Unendlich wachsen, sodass die Weights und Biases nicht zu gross oder zu klein werden.

In dem einzigen Fall wo die Identitätsfunktion $f(x) = x$ benutzt wird und nur ein Input gibt, dann dem Neuron verhaltet sich wie eine lineare Funktion $y = wx + b$.

Die Aktivierungsfunktion ist wichtig in Trainingsprozess, denn sie beeinflusst direkt die Trainingszeit. Es folgt, dass wenn die Aktivierungsfunktion gut gewählt wird, wenige Ressourcen beim Training eingesetzt werden müssen. Auch wenn das Training in Supercomputers stattfindet, werden tiefere Kosten generiert.

3 Das Training

Das sogenannte Training ist der Prozess in welches das neuronale Netzwerk an dem Problem angepasst wird und in welches, dank eine Kostenfunktion alle die Weigts und Biases reguliert werden. Das bis die Kostenfunktion ein globales Minimum erreicht. Die Hauptquelle für dieses Kapitel ist das zweite Kapitel von Python deep Learning (Ivan Vasilev, 2019).

Das Training ein Neuronales Netzwerk ist ein Minimierungsprozzess welche auf $\mathbb{R}^{N+1}$ Dimensionen stattfindet wo $\mathbb{R}^N$ mit $N = B + W$. B ist die Anzahl Biases ist und W die Anzahl Weights.

Im Training wird der tiefste Punkt eine Hyperebene gesucht, Dies fängt an mit zufälligen Werten für alle Variablen. Durch die Berechnung des Gradienten für alle Werte im Trainingsset, werden die Werte soweit angepasst bis der Fehler möglichst klein wird.

Dieses Vorgehen wird Solange wiederholt bis den Fehler nah an 0 liegt; dann werden die trainierten Werte für «W» und »B» als definitiven Parametern der Neuronale Netzwerk gespeichert.

In dieses Teil der Prozess sind sowohl Input als auch Output bekannt. Die unbekannten sind die Parametern W und B. Ein Training Algorithmus macht das gleiche, wie wenn von einem Graphen wo x und y bekannt sind die Funktion $f(x) = y$ bestimmt wird. Wenn $f(x)$eine lineare Funktion ist es sind nur zwei Punkten Nötig, dann je höheres grad die Funktion hat, desto mehr Punkten sind nötig.

Damit dem Training erfolgreich durchgeführt werden kann, wird ein Trainingsset benötigt. Um die Trainingssets, die in diese Arbeit benutzt worden, zu generieren, wurde es die Random Funktion benutzt. Die Trainingssets wurden erstellt in dem erst die beide Vektoren B und W definiert wurden. Dann die drei Parametern R, G und B werden mit zufälligen Werten zwischen 0 und 255 belegt. In reellen Fall werden diese Parameter durch den Sensor vorgegeben. Anhand dieser Werten werden es die Outputs O0 und O1 berechnet und in eine Liste gespeichert, zusammen mit R, G und B. Dieses Weg Trainingssets zu erstellen, ermöglicht nur die Funktionalität des Trainings eines Neuronales Netzwerk zu bestätigen. Diese ist nur einen mathematischen Weg, um zu prüfen, ob das Programm (Python Skript) richtig funktioniert und die Werte von W und B annähern kann.

Das effektive Training geschieht durch das Gradientenverfahren welche später erklärt wird. Das Gradientenverfahren ist ein mathematisches Vorgehen, dem Minimum eine Funktion herausfinden zu können, indem kein Graphen angefertigt werden muss. Dies ist anders als Kurvendiskussion da es allgemeiner ist und auch für komplexere Funktionen funktioniert. Damit dies Vorgehen in der Lage ist, dem Fehler zu minimieren, ist es nötig ein Weg zu finden dem Fehler zu bestimmen. Diese ist die Tätigkeit der Kostenfunktion, welche im nächsten abschnitt erklärt wird.

3.1 Die Kostenfunktion

Die Kostenfunktion ist diejenige Funktion, welche, während das Training sich möglichst dem Wert 0 annähern muss. Somit ist diese Funktion ein Schlüsselfaktor im Training, da damit der Fehler und somit die Qualität des neuronalen Netzes berechnet wird.

Abhängig von Trainingsset: schaut die Kostenfunktion wie gut das NN die gewünschten Daten aus den Trainingsset erhält.

$$C(x) = 0.5|y(x) - \dot{y}|^2$$

Wo y das Output des neuronalen Netzwerkes ist und ẏ das erwartete wert.

Die Formel dieser Funktion ist nicht zufällig da ein klares Minimum hat, wenn $y(x) \in \mathbb{R}^{1-2}$ ist und grafisch einfach zu bestimmen ist. Diese Funktion hat auch eine bequeme Formel für C'(x), da es keine Brüche oder Skalare vorhanden sind, sondern nur $|y(x) - \dot{y}| \cdot y(x)'$; somit sind die Gradienten einfacher zu berechnen.

4 Gradientenverfahren

Das Training durch Gradientenverfahren kann als ein Minimierungsproblem gesehen werden in dem das ganze neuronale Netzwerk mit alle Training Points als eine einzige Funktion C:$\mathbb{R}^n$ ein gesehen wird. Diese Methode wird eingesetzt, wenn eine klassische Kurvendiskussion zu schwierig oder zeitaufwendig wäre.

$$C_t(x_1, x_2 \dots x_n) = \sum_{x=1}^{n} C(T_x) = min$$

Wo $C_t(x)$ die Summe der Kostenfunktion $C(x)$ für alle Training Punkte T, und x die Vektoren der Weights $\vec{w}$ und der Vektor der Biases $\vec{B}$. Die Matrize aller «Training Points» T entählt all die Parametern.

Das Vorgehen des Gradientenverfahren benutzt das Prinzip des rollenden Balls, um dem globalen Minimum zu finden. Diese Analogie hilft eine 3D Vorstellung sich zu machen, wo die Höhe in z-richtung die Summe der Fehler für gegebene Parameter ist. S.Abb.4 Die Ableitung

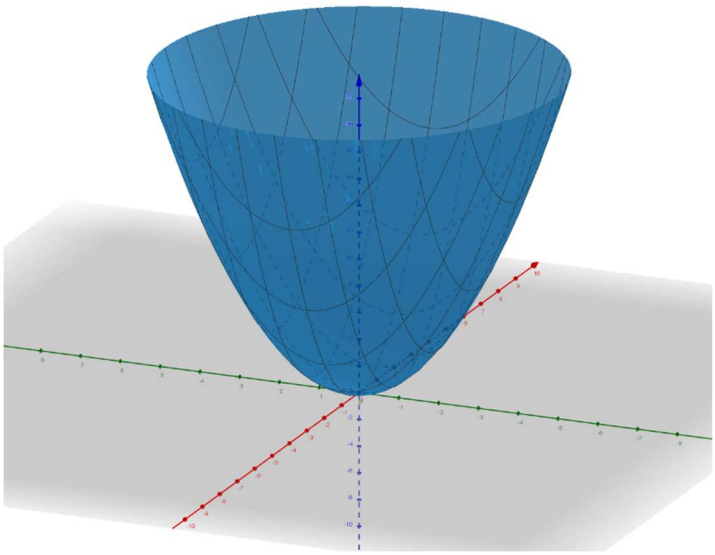


Abbildung 4

der Funktion nach alle ihre Variablen ergibt dem Vektor $\vec{v}$,der Lokale Gradient welcher in die steilste Direktion (nach oben, wenn in 3D) zeigt. Die Komponenten der Vektor sind die Ableitungen nach jeder variabel des Kostenfunktion C

$$\vec{V} = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Allgemein ist dem Gradienten ein Vektorenfeld, wo jeder Vektor in die steilste Richtung für jeder beliebigen Punkt P im Raum zeigt. Der erste Schritt ist den Gradienten zu bestimmen: Um dies zu bestimmen muss man die Funktion nach jeder seiner Variablen ableiten. Der nächste Schritt ist den Gradienten in Punkt P zu bestimmen wie auch dessen Länge. Damit die Korrektion des Gradienten schrittweise geschieht und keine Sprünge gemacht werden, muss die Länge normalisiert werden, indem der Gradient durch seine Länge geteilt wird, sodass die Länge immer 1 ist.

$$\left| \frac{\vec{v}}{|\vec{v}|} \right| = 1$$

Damit die Minimierung Präziser werden kann, wird ein Skalar welches so klein wie möglich sein muss, wird das sogenannte Epsilon ε eingesetzt. Er kann jede beliebige Zahl von $\mathbb{R}^+$, welche die Genauigkeit das Training vorgibt.

$$P_{n+1} = P_n - \varepsilon \frac{\vec{v}}{|\vec{v}|}$$

Eine Möglichkeit, die Minimierung zu beschleunigen, ist folgende: ein variables ε welches am Anfang eine grosse Zahl sein kann und immer kleiner wird bis den Fehler nahezu null erreicht. Falls der neue Fehler grösser werden sollte (als der vorgängige Wert), muss ε künstlich halbiert werden. Dieses Vorgehen wird in dem späteren code in Kap.199 erklärt.

4.1 Beispiel

Jetz wird ein praktisches Beispiel zum Gradientenverfahren illustriert. Dieses Beispiel wird durchgeführt damit den verschiedenen Schritten, das Minimum zu finden durch Gradientrenverfahren, klar werden. Die Funktion, welche für das Beispiel benutzt wird, ist die folgende:

$$C(x,y) = \frac{1}{2}(x-2)^2 + \frac{1}{2}(y-5)^2$$

Der Graph der Funktion, mit seinem eindeutigen Minimum im punkt (2|5) ist in Abb. 5 dargestellt.

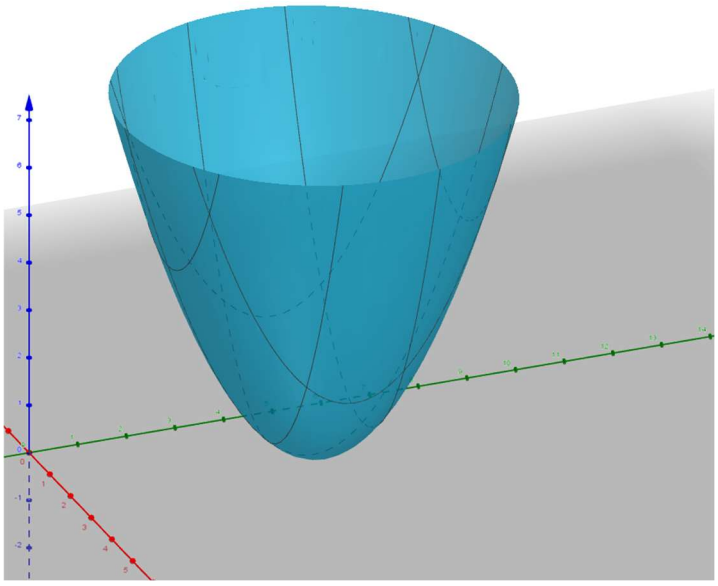


Abbildung 5

Der allgemeine Gradient der Funktion $C(x,y)$ ist:

$$\frac{dC}{dx} = x - 2$$

$$\frac{dC}{dy} = y - 5$$

Für dem Verfahren dieser Funktion es wird das Punkt P_0 als Startpunkt benutzt welche Koordinaten sind $P_0 = (3|3)$ $C(3|3) = 2.5$ und ε=0.5 gewällt. Der Gradient im Punkt P_0 ist

$\begin{pmatrix}1\\-2\end{pmatrix}$ die absolute Länge des Gradienten im P_0 ist $\left|\overrightarrow{v_{P_0}}\right| = 2.236\ldots$ und der Wert $C(P_0)$ ist wie erwähnt 2.5. Die Koordinaten für P_1 werden berechnet indem von P_0 der ε-Wert in Richtung des negativen Gradienten $\nabla C(P_0)$ geht.

$$P_1 = P_0 - \varepsilon \frac{\nabla \mathrm{C}(\mathrm{P}_0)}{|\nabla \mathrm{C}(\mathrm{P}_0)|}$$

Das Resultat dieser Operation ist $P_1 = \begin{pmatrix}2.77\ldots\\3.44\ldots\end{pmatrix}$. Die Werte für x und y müssen in die Funktion $C(x,y)$ eingegeben werden und ergeben das Resultat $C(P_1) = 1.5\ldots$ Für diesen ersten Schritt gab es eine Verbesserung des Fehlers von ~1. Die Fehlerabnahme wird sinken bis nahezu null, weil dem Gradienten kleiner wird da auch dem Fehler kleiner geworden ist. Dem Fehler wird grösser, wenn die Korrektion des Gradienten zu gross ist und dem Minimum überwindet wird, Dieses passiert, weil ε zu gross ist, um dem 0-Punkt zu erreichen.

Der Prozess der Minimierung geht weiter mit den gleichen Schritten bis die Fehler wieder grösser wird (in der Tabelle Zwischen P4 und P5), nach dieses punkt ε muss verkleinert werden damit die Minimierung weitergehen kann.

ε=0.5	P0	P1	P2	P3	P4	P5	P6
x	3	2.7763932	2.5527864	2.32917961	2.10557281	1.88196601	2.10557281
y	3	3.4472136	3.89442719	4.34164079	4.78885438	5.23606798	4.78885438
C	2.5	1.50696601	0.76393202	0.27089803	0.02786405	0.03483006	0.02786405
c'x	1	0.7763932	0.5527864	0.32917961	0.10557281	-0.11803399	0.10557281
c'y	-2	-1.5527864	-1.10557281	-0.65835921	-0.21114562	0.23606798	-0.21114562
|grad(C)|	2.23606798	1.73606798	1.23606798	0.73606798	0.23606798	0.26393202	0.23606798

Tabelle 1

In Tabelle 1 ε bleibt gleich (ε=0.5). Dem Minimum welche erreicht werden kann ist 0.027… Wenn man ε variieren lässt, erhöht sich die Genauigkeit der Minimierung. In der Tabelle 2 ε variiert. Dem Wert von ε verkleinert sich, um ein tieferer Fehler zu erreichen (4.33×10^{-6}).

	P0	P1	P2	P3	P4
x	3	2.5527864	2.10557281	1.99376941	2.00131614
y	3	3.89442719	4.78885438	5.01246118	4.99736772
C	2.5	0.76393202	0.02786405	9.7051E-05	4.3306E-06
ε	1	1	0.25	0.016875	
c'x	1	0.5527864	0.10557281	-0.00623059	0.00131614
c'y	-2	-1.10557281	-0.21114562	0.01246118	-0.00263228
|grad(C)|	2.23606798	1.23606798	0.23606798	0.01393202	0.00294298

Tabelle 2

5 Das Line Following mit LEGO EV3 Roboter

Einer Linie zu folgen, scheint keine schwierige Aufgabe für Menschen zu sein, aber wenn eine Maschine machen muss, ist es anders. Einen Ansatz dieses Problem zu lösen, ist die Nutzung einer oder mehrere Sensoren, um die Linie wahrzunehmen. In einem weiteren Schritt muss einen Algorithmus diese Informationen verarbeiten und die Antriebe steuern. Es ist zu beachten, dass je mehr Rohinformationen (durch eine Vielzahl an Sensoren aufgenommen) vorhanden sind, desto "einfacher" wird die Aufgabe. Demzufolge wird die Programmierung zum Teil einfacher. Das Line Following (einer Linie auf dem Boden folgen) erscheint als «nutzloses» Problem. Es ist aber eine Grunbdaustein zur Lösung des Problems der selbststeuernden Autos, welche auf einer Spur bleiben und dieser folgen müssen. Von disem Punkt bis zum autonomen Fahren ist es aber noch ein langer Weg. Mit einer Vielzahl weitere, zum Teil ähnlicher Aufgaben.

Die Quellen dieses Fünftes Kapitel sind die eigene Veruschserfahrung, die Webseite von ev3dev (ev3dev, 2023) und dem Blog von J. Sluka auf die Webseite inpharmix.com (Sluka, 2009)

5.1 Der Roboter

Die Lego Plattform EV3 ist eine von LEGO entwickelte Plattform, um die neuen Generationen in die Welt der Informatik und der Robotik spielerisch einzuführen (LEGO, 2019). Diese Plattform hat verschiedene Schwierigkeitsstufen, welche verschiedene Betriebssysteme nutzen. Das erlaubt ein erleichterter Lernprozess für dem Lernende. Die einfachste Plattform ist die Lego EV3 App, mit welcher einfache Programme zusammengestellt werden können, in dem man vorprogrammierte Module zusammenbastelt und dann auf dem Roboter per Bluetooth hochladet und laufen lässt. Auf der nächsten Stufe kann man direkt in Jython[1] programmieren und durch eine WiFi Verbindung die Programme auf den Roboter hochladen. Danach kann der Roboter das Programm selbst ausführen. Diese Methode erlaubt mehr Freiheit und Komplexität in den Programmen. Eine weitere Stufe welche noch mehr Freiheit und Zugriff zu allen Modulen von Python (und C) bietet, ist das EV3DEV- System. In dieser Arbeit werde ich auf dem Roboter die Plattform ev3dev benutzen und als IDE[2] den Visual Studio Code mit der Extension ev3dev -Browser benutzen. Dieses System ist eine Open Source Plattform, welche das Lego EV3 durch eine Micro SD mit einem anderen Betriebssystem steuert. Somit sind das Programmieren und Skripten in verschiedenen Programmiersprachen möglich.

[1] Java Python ist eine Untersprache von Python, welche auf Java programmiert ist.
[2] En: Integrated Development Environment

5.2 Die Installation des Betriebssystems

Um ev3dev auf einem Ev3 Roboter laufen zu lassen braucht man zuerst eine Micro SD, auf welcher man das Image des Betriebssystems brennen muss.

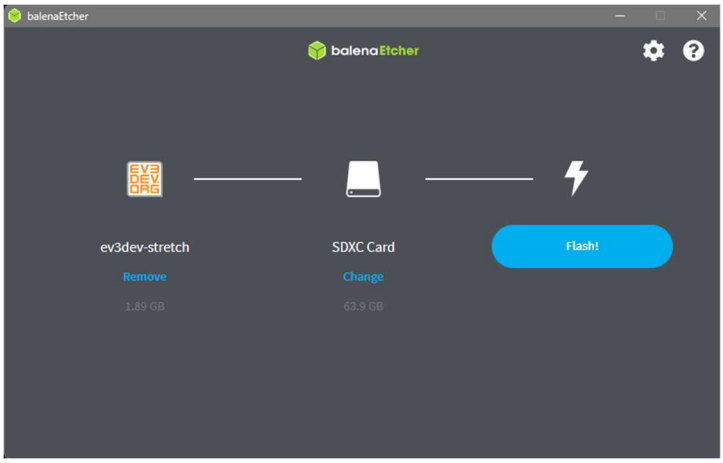


Abbildung 6

So gehet man vor:

1. Schritt: Distro [3]Image von ev3dev vom ”GitHub Repostory” von ev3dev herunterladen und mit einem beliebigen Programm entpacken.

2. Schritt: Um das Image auf die SD-Karte zu laden, braucht man Etcher (balena Etcher), welche von der Projektwebseite (balena, 2023) heruntergeladen werden kann.

3. Schritt: Nachdem die ganze Software vorhanden ist und die SD-Karte in den Rechner eingesteckt ist, muss man Etcher öffnen und das File mit dem Suffix «.iso» auswählen. S. Abb.6

4. Schritt: Nun folgt die Wahl des Gerätes, auf welches das Image gebrannt werden soll. Es muss beachtet werden, dass das richtige Gerät gewählt wird. Ansonsten werden alle Dateien auf ein falsches Gerät überschrieben.

5. Schritt: Nachdem das Gerät gewählt wurde, muss die Schalfläche «Flash» angeklickt werden, damit die Software auf die SD-Karte geschrieben werden kann.

6. Schritt: Wenn die SD-Karte vorhanden ist, ist nur noch der System Boot auf dem LEGO EV3 Brick zu machen. Dafür muss man die SD-Karte im SD-Laufwerk des Roboters einsetzen und den Roboter einschalten. Der erste Boot kann bis zu 10 Minuten brauchen, die späteren Boots benötigen weniger Zeit.

7. Schritt: Nun muss man die Verbindung des Roboters zu eine Wireless Local Network (WLAN) herstellen. Am besten eins ohne Internet-Verbindung, damit keine Sicherheitslücke geöffnet wird.

[3] Linux Distribution

8. Schritt: Nachdem das Gerät «online» ist, wird eine IP-Adresse gezeigt, welche den Roboter auf dem Netzwerk identifiziert und dann die Verbindung mit dem Roboter ermöglicht.

9. Wenn die zwei Maschinen (Computer und Roboter) durch WLAN verbunden sind, ist es möglich, die Python Scripts auf den Roboter herunterzuladen und laufen zu lassen oder ihn mit einer «integrated" command line zu verknüpfen, welche eine direkte Interaktion mit dem Roboter ermöglicht.

Diese ist eine allgemeine Installationsvorgehen aber für eine ausführlichere Erklärung, es gibt die Webseite von ev3dv (ev3dev, 2023) Welche mehr Details hat.

5.3 Lösungsansatz mittels klassischer Regeltechnik

Damit der Roboter einer Linie folgen kann, benötigt er zwei grosse Motoren für die Steuerung der Räder, welche die Richtung vorgeben und ein Lichtsensor, welche die Reflektion der Oberfläche misst S.Abb.7 (Sluka, 2009) Der einfachste Weg damit ein Roboter einer Linie folgen kann – aber nur mit minimaler Genauigkeit ist «ein Loop» mit einem «if» Statement. Wenn die Reflektion grösser als ein vorbestimmter Wert (x) ist, muss der Roboter nach rechts abbiegen, wenn kleiner nach links. Diese Version funktioniert, aber ist sehr ungenau. Der Roboter fährt immer auf die eine oder

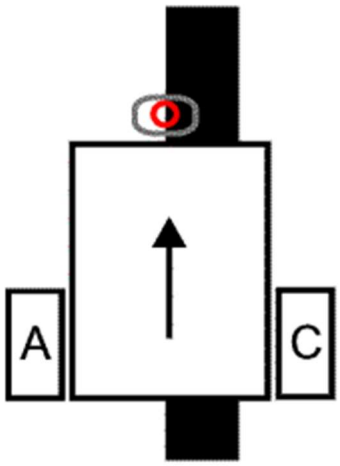


Abbildung 7 (Sluka, 2009)

```python
#!/usr/bin/env python3

from ev3dev2 import*
from ev3dev2.motor import LargeMotor,OUTPUT_A,OUTPUT_B, SpeedPercent
from ev3dev2.sensor import INPUT_2
from ev3dev2.sensor.lego import ColorSensor
#TODO:
rm= LargeMotor(OUTPUT_A)
lm= LargeMotor(OUTPUT_B)
lf=ColorSensor(INPUT_2)
x=5

while True:
    value=lf.reflected_light_intensity
    if value> x:
        lm.on(SpeedPercent(20))
        rm.on(SpeedPercent(-20))
    else:
        lm.on(SpeedPercent(-20))
        rm.on(SpeedPercent(20))
```

die andere Seite der Linie; bei engen Kurven kann die Maschine die Linie verlieren und diese dann nicht mehr finden. Ein besserer Algorithmus ist derjenige, wo der Roboter, wenn der Reflektion Index genau gleich wie der gewünschte Wert ist, geradeaus fährt. Diese Version ist besser, aber hat das gleiche Problem wie die Erste. Eine bessere Version ist die Proportionale, bei welcher mit zunehmendem Fehler auch die Geschwindigkeit der Korrektion zunimmt, in beide positive und negative Richtungen. Diese letzte Methode ist gut aber viel zu reaktiv. Diese zu hohe Reaktivität führt zu einem

```python
while True:
    value=lf.reflected_light_intensity
    if value> x:
        lm.on(SpeedPercent(20))
        rm.on(SpeedPercent(-20))
    elif value==x:
        lm.on(SpeedPercent(20))
        rm.on(SpeedPercent(20))
    else:
        lm.on(SpeedPercent(-20))
        rm.on(SpeedPercent(20))
```

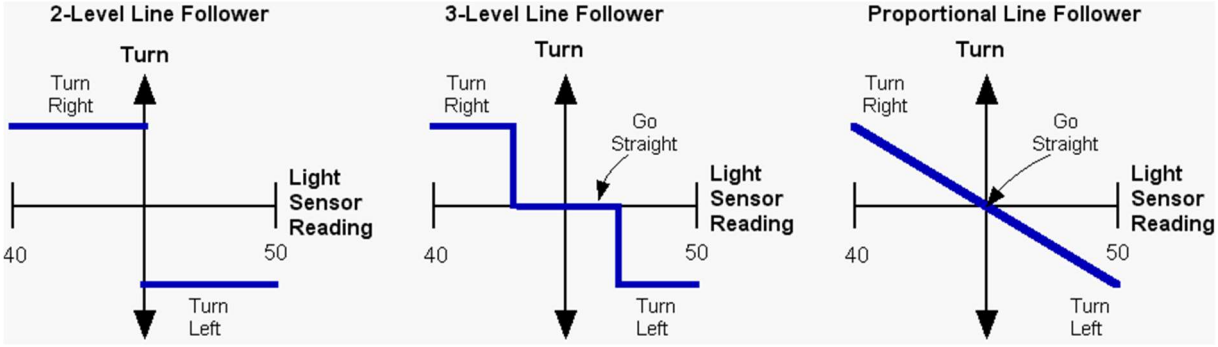


Abbildung 8 (Sluka, 2009)

Verlust der Linie bei engen Kurven und einer allgemeinen Nervosität des Roboters während des ganzen line following Prozesses. Alle diese Methoden und Algorithmen haben gemeinsam, dass sie sich nicht erinnern, was sie früher gemacht haben und reagieren deshalb gleichzeitig zu den geänderten Bedingungen, die der Sensor erkennt. Das führt zu einer Überaktivität im Normalfall aber zu einer Unterreaktivität im Extremfall, wie enge Kurven oder Kanten.

Eine mögliche Lösung ist der sogenannte PID-Controller (English: Proportional Integral Derivative). Hier wird der Roboter erst proportional kontrolliert, in einen weiteren Schritt wird die Maschine integral kontrolliert, das heisst den Fehler wird aufsummiert. Wenn ein Fehler für lange Zeit bestehet, wird die Integral Grösser dementsprechend Stark korrigiert. Das PID-Controller funktioniert gut in fast alle Situationen, Aber die drei Skalaren, die für Proportional, Integral und Differential stehen, sind schwierig manuell zu Kalibrieren. Ein Problem diese Werte von Hand zu kalibrieren, ist die Schwierigkeit, Live-Verbesserungen zu sehen und die Skalar Änderungen auf dem Roboter während dem laufenden Programm zu übergeben, zusätzlich hängen diese Werten voneinander.

```python
#!/usr/bin/env python3

from time import*
from ev3dev2 import*
from ev3dev2.motor import LargeMotor,OUTPUT_A,OUTPUT_B, SpeedRPM
from ev3dev2.sensor import INPUT_2
from ev3dev2.sensor.lego import ColorSensor

#TODO:
rm= LargeMotor(OUTPUT_A)
lm= LargeMotor(OUTPUT_B)
lf=ColorSensor(INPUT_2)
trgt=332.25
print(trgt)

pg=0.2500 #kp error
ig=0.01125#ki errsum
dg=0.2 #kd errdiff

inte=0
lr=0
erh=[0.0,0.0,0.0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
v=50
history=[]
lf.raw
while True:
    c=sum(lf.raw)
    error=sum(lf.raw)-trgt
    erh[1:19]=erh[0:18]
    erh[0]=error
    inte=sum(erh[0:19])
    der=error-erh[1]
    rot=pg*error+ig*inte+dg*der
    lm.on(SpeedRPM(v+rot))
    rm.on(SpeedRPM(v-rot))
```

5.4 Ist das eine Aufgabe ein neuronales Netzwerk?

Kann ein Neuronales Netzwerk diese Schwierigkeiten lösen? Die Idee hinter Neuronalen Netzwerken ist es, schwierige Probleme zu lösen, für welche eine «Hard Coded» Variante zu schwierig wäre oder komplett unmöglich ist. Ein Beispiel ist das Problem der Kalibrierung zu lösen. Das Neuronale Netzwerk, welches in dieser Arbeit getestet wird, hat als Input ein einziger Lichtsensor, welcher Dateien für die drei Farben R, G und B zwischen 0 und 255

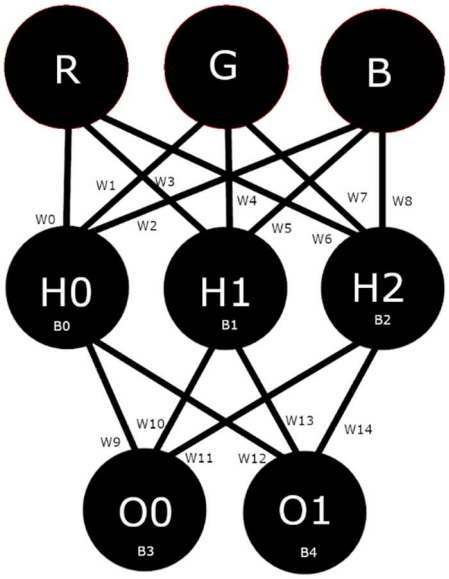


Abbildung 9

ausgibt, drei Neuronen sind im Hidden Layer H0, H1 und H2 und zwei Neuronen im Output Layer O0 und O1, welche die Geschwindigkeit für die zwei Motoren ausgeben.

6 Ansatz der Lösung des Problems «Linefollowing» mit einem neuronalen Netz.

Abb. 10 zeigt ein Beispiel eines neuronalen Netzwerks, welche für dieses Arbeit benutzt wurde. Das ist ein neuronales Netzwerk mit tiefe 3 und mit drei Inputneuronen: R G und B, drei im Hiddenlayer: H0, H1 und H2 und Zwei im Outputlayer: O0 und O1.

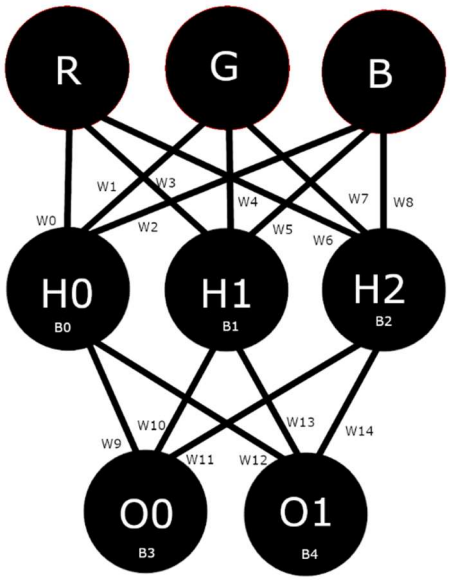


Abbildung 10

Diese ist eine einfache feedforward Neural Network, wo die drei Inputwerte: R, G und B für dem Neuron H0 an den Weights W0, W1 und W2 multipliziert werden und dann miteinander summiert zusammen mit B0 bevor das Ganze in die Aktivierungsfunktion reingetan wird und H0 ausgibt. Das gleiche gilt für die Neuronen H1 und H2 mit den Weights W3, W4, W5 und B1 für H1 und mit H6, H7, H8 und B2. Dieser Prozess ist wie ein Vektor (1x3[R, G, B]) Matrize (3x3[W0-8]) Multiplikation und eine Adduktion von ein Zweites Vektor (1x3[B0-2]) und als Resultat folgt dem Vektor 1x3 H0, H1 und H2. Ähnlich passiert für die Berechnung von O0 und O1. Für O0 es werden H0, H1, und H2 an den Weights W9, W10 und W11 multipliziert und es werden wie früher schon summiert zusammen auch mit B3. Das Gleiche gilt für O1 mit W12, W13, W14 und B4. Dieses Vorgehen ist wie früher ein Vektor (1x3 [H0-2]) welche mit einer Matrize (3x2 [W9-14]) multipliziert wird und mit dem Vektor (1x2 [B3, B4]) summiert wird

7 Funktionen und Ableitungen

In diesem Kapitel werden alle Formeln erst für die neuronale Netzwerk und dann für dem Gradient des Kostenfunktion beschrieben.

Die Aktivierungsfunktion ist $f(x) = \frac{1}{1+e^{-x}}$ und ihre Ableitung $f'(x) = \frac{e^{-x}}{(1+e^{-x})^2}$ bleiben durch das ganze neuronale Netzwerk unverändert.

Das erste Element der Hiddenlayer H0 hat der form $H0 = f(Rw_0 + Gw_1 + Bw_2 + B_0)$ ähnlich für $H1 = f(Rw_3 + Gw_4 + Bw_5 + B_1)$und für das letzte Neuron $H2 = f(Rw_6 + Gw_7 + Bw_8 + B_2)$. Diese drei Neuronen haben alle die gleiche Form; erst die Summe der Multiplikation von Inputs mit den Weights mit dann dem Bias, Die Summe wird dann In die Aktivierungsfunktion eingesetzt.

Die Outputlayer hat nur zwei Neuronen, welche die Geschwindigkeit der Motoren vorgeben. Die Inputwerte für beide Neuronen in dem letzten Layer sind die Outputs der Neuronen im Hidden Layer. Die Formeln sind: $O0 = f(H0w_9 + H1w_{10} + H2w_{11} + B_3)$ und $O1 = f(H0w_{12} + H1w_{13} + H2w_{14} + B_4)$. Von diesen Formeln kann man erkennen, dass O0 und O1 von R, G und B sowie die Zwei Vektoren $\vec{W}$ und $\vec{B}$ abhängen.

Damit dem Training erfolgreich durchgeführt werden kann muss den Fehler berechnet werden. Der Fehler in so ein neuronales Netzwerk wird, mit der Abweichung der berechneten Werten mit den gewünschten berechnet. Die Kostenfunktion für diese Neuronale Netzwerk ist: $C(\vec{W}, \vec{B}) = \frac{1}{2}(O0 - \exp0)^2 + \frac{1}{2}(O1 - \exp1)^2$ welche ihre Ableitung: $C'(\vec{W}, \vec{B}) = (O0 - exp0)(O0)' + (O1 - \exp1)(O1)'$ist. Aus dieser Ableitung folgt dem Gradeienten von C, welche 20-dimensional ist.

$$\nabla C$$

$$\frac{dC}{dw_0} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)\big(f'(Rw_0 + Gw_1 + Bw_2 + b_0)(R)\big)w_9 + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_0 + Gw_1 + Bw_2 + b_0)(R)\big)w_{12}$$

$$\frac{dC}{dw_1} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)\big(f'(Rw_0 + Gw_1 + Bw_2 + b_0)(G)\big)w_9 + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_0 + Gw_1 + Bw_2 + b_0)(G)\big)w_{12}$$

$$\frac{dC}{dw_2} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)\big(f'(Rw_0 + Gw_1 + Bw_2 + b_0)(B)\big)w_9 + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_0 + Gw_1 + Bw_2 + b_0)(B)\big)w_{12}$$

$$\frac{dC}{dw_3} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)\big(f'(Rw_3 + Gw_4 + Bw_5 + b_1)(R)\big)w_{10} + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_3 + Gw_4 + Bw_5 + b_1)(R)\big)w_{13}$$

$$\frac{dC}{dw_4} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)(f'(Rw_3 + Gw_4 + Bw_5 + b_1)(G))w_{10} + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_3 + Gw_4 + Bw_5 + b_1)(G)\big)w_{13}$$

$$\frac{dC}{dw_5} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)(f'(Rw_3 + Gw_4 + Bw_5 + b_1)(B))w_{10} + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'Rw_3 + Gw_4 + Bw_5 + b_1)(B)\big)w_{13}$$

$$\frac{dC}{dw_6} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)(f'(Rw_6 + Gw_7 + Bw_8 + b_2)(R))w_{11} + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_6 + Gw_7 + Bw_8 + b_2)(R)\big)w_{14}$$

$$\frac{dC}{dw_7} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)(f'(Rw_6 + Gw_7 + Bw_8 + b_2)(G))w_{11} + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_6 + Gw_7 + Bw_8 + b_2)(G)\big)w_{14}$$

$$\frac{dC}{dw_8} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)(f'(Rw_6 + Gw_7 + Bw_8 + b_2)(B))w_{11} + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_6 + Gw_7 + Bw_8 + b_2)(B)\big)w_{14}$$

$$\frac{dC}{dw_9} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)H0 + 0$$

$$\frac{dC}{dw_{10}} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)H1 + 0$$

$$\frac{dC}{dw_{11}} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)H2 + 0$$

$$\frac{dC}{dw_{12}} = 0 + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)H0$$

$$\frac{dC}{dw_{13}} = 0 + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)H1$$

$$\frac{dC}{dw_{14}} = 0 + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)H2$$

$$\frac{dC}{db_0} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)(f'(Rw_0 + Gw_1 + Bw_2 + b_0)(w_9)) + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_0 + Gw_1 + Bw_2 + b_0)(w_{12})\big)$$

$$\frac{dC}{db_1} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)(f'(Rw_3 + Gw_4 + Bw_5 + b_1)(w_{10})) + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_3 + Gw_4 + Bw_5 + b_1)(w_{13})\big)$$

$$\frac{dC}{db_2} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)(f'(Rw_6 + Gw_7 + Bw_8 + b_2)(w_{11})) + (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)\big(f'(Rw_6 + Gw_7 + Bw_8 + b_2)(w_{14})\big)$$

$$\frac{dC}{db_3} = (O0 - exp0)f'(H0w_9 + H1w_{10} + H2w_{11} + b_3)$$

$$\frac{dC}{db_4} = (O1 - \exp1)f'(H0w_{12} + H1w_{13} + H2w_{14} + b_4)$$

Diese Formeln wurden vereinfacht, indem den Neuronen der Hiddenlayer (H0, H1 und H2) Direkt in der Formel eingesetzt werden.

8 Programmcode auf dem Roboter

In dieses Kapitel werden die selbst programmierten Python Skripts vorgestellt, welche der Roboter Steuern und den Vektoren W und B bestimmen.

8.1 Neuronales Netzwerk Für LEGO-Roboter

Das Programm, welches am Ende auf dem Roboter läuft, ist nur das Neuronale Netzwerk welche den Outputs O0 und O1 berechnet. Dieses Schlankes Programm läuft ressourcensparend auf dem Roboter, der nicht mit einem leistungsfähigen Rechner ausgerüstet werden muss.

```
1  #!/usr/bin/env python3
2  from math import exp
3  from ev3dev2 import*
4  from ev3dev2.motor import LargeMotor,OUTPUT_A,OUTPUT_B, SpeedPercent
5  from ev3dev2.sensor import INPUT_2
6  from ev3dev2.sensor.lego import ColorSensor
7
8  #TODO:
9  rm= LargeMotor(OUTPUT_A)
10 lm= LargeMotor(OUTPUT_B)
11 lf=ColorSensor(INPUT_2)
12
13 W=[ 0.22761073,0.31168518,0.98887353,0.41346739,0.28467911,0.76328284,0.62464784,0.44168643,
14       0.17191695,0.01403663,-0.40239056,-0.12775812,-0.6416017 ,-0.23089563,0.13047133]
15 B=[ 0.99682834,0.79983418,0.37870556,-0.17703514,0.04887882]
16
17 def f(x):
18     return 1/(1+exp(-x))
```

Die Erste Linie dieses Codes sagt dem Roboter wie dem Python Skript ausgeführt werden muss. Dann werden es die nötigen Python Libraries importiert, die Math Library für dem Eulersche Zahl und die ev3dev2 Library damit dem Roboter den Sensor lesen und die Motoren steuern kann.

Der eigentliche Code fängt erst nach dem #TODO: in Zeile 8 an. Die Globalen Variablen werden initialisiert. Es werden die Variablen definiert und auch zu den Klassen der zwei Motoren und des Lichtsensores zugewiesen.

```
20  def H0pre(inp,W,B):
21      return inp[0]*W[0]+inp[1]*W[1]+inp[2]*W[2]+B[0]
22
23  def H0(inp,W,B):
24      return f(H0pre(inp,W,B))
25
26  def H1pre(inp,W,B):
27      return inp[0]*W[3]+inp[1]*W[4]+inp[2]*W[5]+B[1]
28
29  def H1(inp,W,B):
30      return f(H1pre(inp,W,B))
31
32  def H2pre(inp,W,B):
33      return inp[0]*W[6]+inp[1]*W[7]+inp[2]*W[8]+B[2]
34
35  def H2(inp,W,B):
36      return f(H2pre(inp,W,B))
37
38  def O0pre(inp,W,B):
39      return float(H0(inp,W,B))*W[9]+float(H1(inp,W,B))*W[10]+float(H2(inp,W,B))*W[11]+B[3]
40
41  def O0(inp,W,B):
42      return f(O0pre(inp,W,B))
43
44  def O1pre(inp,W,B):
45      return float(H0(inp,W,B))*W[12]+float(H1(inp,W,B))*W[13]+float(H2(inp,W,B))*W[14]+B[4]
46
47  def O1(inp,W,B):
48      return f(O1pre(inp,W,B))
49
50  def evalnn(inp,W,B):
51      re=O0(inp,W,B)
52      li=O1(inp,W,B)
53      return re,li
54  while True:
55      inp=lf.rgb
56      r,l=evalnn(inp,W,B)
57      rm.on(SpeedPercent(r*10))
58      lm.on(SpeedPercent(l*10))
```

Nächstens werden die Parametern W für den 15 Weights und B für den 5 Biases definiert. Dieser Code ist modular aufgebaut damit das Neuronale Netzwerk einfach zu korrigieren ist und damit das Ganze verständlicher wird.

Mit der Definition f(x) es wird die Aktivierungsfunktion, welche dann in den anderen Befehlen benutzt wird, definiert.

Von H0pre in Linie 20 bis zur Definition O1 in Linie 48 werden die Neuronen definiert. Jedes Neuron ist durch zwei Funktionen definiert. Die erste für die Summe der Inputs mal Weights plus Bias und die zweite für die Berechnung nach Anwendung des Aktivierungsfunktion.

Jede Funktion hat als Parameter Inp: das Input (R, G und B) welche direkt von dem Lichtsensor aufgenommen wird und die zwei Vektoren W und B welche die Parametern der Weights und der Biases definieren. Die letzte Definition (evalnn) schliesst das neuronale Netzwerk ab und gibt den werten für Re(O0) und Li(O1) zurück welche die prozentualen Geschwindigkeiten der Motoren (rechts und links) entsprechen.

8.2 Ausführung des Programmes

In die Letzte fünf Linien des Skriptes fängt das Programm zu rechnen an. Dem while loop am Ende ermöglicht dem Programm zeitlich unbeschränkt zu laufen. Einen Abbruch kann durch die Berührung einen Touchsensoren eingeleitet werden. In die 55. Linie misst der Lichtsensor die Reflektion der Unterlage damit die Werten in die Neuronale Netzwerk eingegeben werden können. In die Linie 56 werden es die Werte für r und l berechnet, welche die Geschwindigkeiten der zwei Motoren vorgeben und in der Linie 57-8 werden die Befehle an den Motoren abgegeben.

9 Training des NN auf eigenen Rechner

In Dieses Kapitel wird der Python-code erklärt, welcher für das Training zuständig ist.

```
1  from math import exp
2  import numpy as np
3  import matplotlib.pylab as plt
4  import keyboard
5
6  Wg=np.random.rand(15,1)
7  Bg=np.random.rand(5,1)
8
9  tset4=[[46.14942747766261, 247.36208389705283, 74.08030744641174, 0.81659393, 0.91809978],[133.33315332807823, 117.71784034715313, 108.35543734187672, 0.81659393, 0.91809978],
10       [197.66346409625467, 183.56815506745406, 134.3071830935232, 0.81659393, 0.91809978],[189.02759701085495, 82.06516146029526, 49.941384305976115, 0.81659393, 0.91809978],
11       [207.77045285493006, 28.652714608810204, 117.00318360637586, 0.81659393, 0.91809978],[217.06198467287302, 254.42309173931255, 113.27118072812857, 0.81659393, 0.91809978],
12       [200.69802942376225, 150.14609728332036, 24.68228158639408, 0.81659393, 0.91809978],[104.50758209407022, 213.72096341760204, 80.64609595214219, 0.81659393, 0.91809978],
13       [234.60614035888858, 92.50591927648051, 169.84191157618739, 0.81659393, 0.91809978],[192.15905854374887, 23.788168811017936, 152.64503882316762, 0.81659393, 0.91809978],
14       [135.02355074609744, 115.45833034239008, 125.95283619782644, 0.81659393, 0.91809978],[215.5057541073719, 221.50080521570374, 83.85481371566169, 0.81659393, 0.91809978],
15       [183.52959442689735, 200.9138247697748, 135.5292693627817, 0.81659393, 0.91809978],[195.34023292787631, 123.34621893684788, 107.18389635903542, 0.81659393, 0.91809978],
16       [179.6591004825346, 201.39125956349358, 241.52247183974075, 0.81659393, 0.91809978]]
17
18 def f(x):
19     return 1/(1+exp(-x))
20
21 def fder(x):
22     return (exp(-x))/((1+exp(-x))**2)
23 def H0pre(inp,W,B):
24     return inp[0]*W[0]+inp[1]*W[1]+inp[2]*W[2]+B[0]
25
26 def H0(inp,W,B):
27     return f(H0pre(inp,W,B))
28
29 def H1pre(inp,W,B):
30     return inp[0]*W[3]+inp[1]*W[4]+inp[2]*W[5]+B[1]
31
32 def H1(inp,W,B):
33     return f(H1pre(inp,W,B))
34
35 def H2pre(inp,W,B):
36     return inp[0]*W[6]+inp[1]*W[7]+inp[2]*W[8]+B[2]
37
38 def H2(inp,W,B):
39     return f(H2pre(inp,W,B))
40
41 def O0pre(inp,W,B):
42     return float(H0(inp,W,B))*W[9]+float(H1(inp,W,B))*W[10]+float(H2(inp,W,B))*W[11]+B[3]
43
44 def O0(inp,W,B):
45     return f(O0pre(inp,W,B))
46
47 def O1pre(inp,W,B):
48     return float(H0(inp,W,B))*W[12]+float(H1(inp,W,B))*W[13]+float(H2(inp,W,B))*W[14]+B[4]
49
50 def O1(inp,W,B):
51     return f(O1pre(inp,W,B))
52
53 def evalnn(inp,W,B):
54     re=O0(inp,W,B)
55     li=O1(inp,W,B)
56     return re,li
57
58 def c(tval,W,B):
59     re,li=evalnn(tval[0:3],W,B)
60     return 0.5*(re-tval[3])**2+0.5*(li-tval[4])**2
61
62 def ctt(tval,W,B):
63     ct=0
64     for val in tval:
65         ct+=c(val,W,B)
66     return ct
67
68 def gradc(tval,W,B):
69     inp=[0.0,0.0,0.0]
70     inp[0:3]=tval[0:3]
71     ret=tval[3]
72     lit=tval[4]
73     gradW=np.zeros((15,1))
74     gradB=np.zeros((5,1))
75     re,li=evalnn(tval[0:3],W,B)
76
```

9.1 Neuronales Netz

Die ersten Zeilen dieses Programm dienen zu dem Import des Libraries und zur Definition der globalen Variablen. In diesem Fall Wg und Bg sind zufällige Vektoren, mit welchen den Weights und Biases initialisiert werden. Tset4 ist dem Trainingsset welche 15 Elementen hat. Jedes Element beinhaltet eine Liste mit 5 werten: R, G, B und die erwarteten Werten für rechts(O0) und links(O1).

f(x) und fder(x) sind die Aktivierungsfunktion und ihre Ableitung, welche in dem Gradienten vorkommen. Zwischen Zeile 22 und 50 werden es die Neuronen definiert mit All ihre Abhängigkeiten. Dieser Teil des Codes wird nicht nochmals erläutert da genau das gleiche ist, wie im Kapitel 8

Diese Definitionen der Neurone rufen sich gegenseitig auf. Die beide letzte Neuronen rufen die Neuronen der Hidden Layer auf und diese die Inputs welche als Parameter mitgegeben werden.

Die Funktion evalnn in Zeile 53 bis 56 ist diejenige Funktion, welche dem Output des neuronalen Netzwerks sammelt und ausgibt. Diese Funktion wird mehrmals verwendet, immer wenn die Bestimmung der Outputs des Neuronale Netzwerk nötig ist.

9.2 Kostenfunktion

Nach der Definition des Neuronalen Netzwerk wird die Kostenfunktion definiert. Die Ist: $C(\vec{W},\vec{B}) = \frac{1}{2}(O0 - \exp0)^2 + \frac{1}{2}(O1 - \exp1)^2$

Anders als in den früheren Funktionen sind den Parametern, welche nicht nur R, G und B enthalten, sondern das ganze Trainigspunkt des Trainingsset. In dieser sind auch die zwei Werten exp0 und exp1 gespeichert. Diese zwei wichtige Werte sind als tval[3] und tval[4] gespeichert, da tval[0:3] (0, 1, 2) sind die drei Parametern für R, G und B. Die Werten für W und B werden als Parametern übertragen damit die globalen Variablen nicht immer gewechselt werden müssen und damit das Programm Flexibler ist.

Die nächste Funktion ist ctt mit den Parametern tval, W und B. Diese Funktion hat ein for loop drinnen, welche den Fehler für jeder Trainingspunkt summiert und am Ende des Prozesses (wenn dem Fehler von jeder Training Point summiert wurde) die Summe ausgibt.

9.3 Gradienten der Kostenfunktion

Damit dem Training stattfinden kann, ist es nötig das Gradient von C zu berechnen. Den Parametern von gradc sind tval: (die Werte ein Trainingspunkt mit den erwarteten Resultaten) und W sowie B. In der Funktion wird zuerst die lokale Variabel inp initialisiert und dann werden die daten R, G und B von der Trainingsset in die Variabel Inp gespeichert. Als zweiter Schritt werden es zwei andere Variablen initialisiert: GradW und GradB welche zwei Nullvektoren sind. Drittens werden es re und li berechnet, indem evalnn aufgerufen wird. Danach wird es dem Gradienten berechnet nach der Formel von Kap7. Am Ende gibt das Befehl Gradw und GradB zurück.

```
77      gradW[0]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[0]*W[9]  +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[0]*W[12]
78      gradW[1]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[1]*W[9]  +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[1]*W[12]
79      gradW[2]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[2]*W[9]  +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[2]*W[12]
80      gradW[3]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[0]*W[10] +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[0]*W[13]
81      gradW[4]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[1]*W[10] +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[1]*W[13]
82      gradW[5]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[2]*W[10] +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[2]*W[13]
83      gradW[6]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[0]*W[11] +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[0]*W[14]
84      gradW[7]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[1]*W[11] +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[1]*W[14]
85      gradW[8]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[2]*W[11] +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[2]*W[14]
86
87      gradW[9]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(H0(inp,W,B))
88      gradW[10]=(re-ret)*float(fder(O0pre(inp,W,B)))*float(H1(inp,W,B))
89      gradW[11]=(re-ret)*float(fder(O0pre(inp,W,B)))*float(H2(inp,W,B))
90      gradW[12]=(li-lit)*float(fder(O1pre(inp,W,B)))*float(H0(inp,W,B))
91      gradW[13]=(li-lit)*float(fder(O1pre(inp,W,B)))*float(H1(inp,W,B))
92      gradW[14]=(li-lit)*float(fder(O1pre(inp,W,B)))*float(H2(inp,W,B))
93
94      gradB[0]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*W[9] +(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*W[12]
95      gradB[1]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*W[10]+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*W[13]
96      gradB[2]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*W[11]+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*W[14]
97      gradB[3]= (re-ret)*float(fder(O0pre(inp,W,B)))
98      gradB[4]= (li-lit)*float(fder(O1pre(inp,W,B)))
99      return gradW, gradB
```

In der Linien 101 bis 108 wird es die Funktion gradctot programmiert. Diese Funktion summiert den Gradienten aller Training Points in zwei Vektoren zusammen, ein für W und das zweite für B.

9.4 Update der Weights und Biases

```
100
101 def gradctot(tval,W,B):
102     gradWtot=np.zeros((15,1))
103     gradBtot=np.zeros((5,1))
104     for i in tval :
105         gradW, gradB =gradc(i,W,B)
106         gradWtot+=gradW
107         gradBtot+=gradB
108     return gradWtot, gradBtot
109
110 def adjustment(eps,Wadj,Badj):
111     global Wg,Bg
112     W=np.zeros((15,1))
113     B=np.zeros((5,1))
114     tot=np.zeros((20,1))
115
116     tot[0:15]=Wadj
117     tot[15:20]=Badj
118     lengh=np.linalg.norm(tot)
119     W[0]=Wg[0]-(Wadj[0]/lengh)*eps
120     W[1]=Wg[1]-(Wadj[1]/lengh)*eps
121     W[2]=Wg[2]-(Wadj[2]/lengh)*eps
122     W[3]=Wg[3]-(Wadj[3]/lengh)*eps
123     W[4]=  Wg[4]-(Wadj[4]/lengh)*eps
124     W[5]=  Wg[5]-(Wadj[5]/lengh)*eps
125     W[6]=  Wg[6]-(Wadj[6]/lengh)*eps
126     W[7]=  Wg[7]-(Wadj[7]/lengh)*eps
127     W[8]=  Wg[8]-(Wadj[8]/lengh)*eps
128     W[9]=  Wg[9]-(Wadj[9]/lengh)*eps
129     W[10]=Wg[10]-(Wadj[10]/lengh)*eps
130     W[11]=Wg[11]-(Wadj[11]/lengh)*eps
131     W[12]=Wg[12]-(Wadj[12]/lengh)*eps
132     W[13]=Wg[13]-(Wadj[13]/lengh)*eps
133     W[14]=Wg[14]-(Wadj[14]/lengh)*eps
134     B[0]=  Bg[0]-(Badj[0]/lengh)*eps
135     B[1]=  Bg[1]-(Badj[2]/lengh)*eps
136     B[2]=  Bg[2]-(Badj[2]/lengh)*eps
137     B[3]=  Bg[3]-(Badj[3]/lengh)*eps
138     B[4]=  Bg[4]-(Badj[4]/lengh)*eps
139     middelta=(np.mean(Wg-W)+np.mean(Bg-B))/2
140     return W,B, middelta, lengh
141
142 def train(set):
143     global Wg,Bg
144     W=np.zeros((15,1))
145     B=np.zeros((5,1))
146     leng=None
147     change=None
148     n=0
149     eps=1
150     H=[]
151     errold=ctt(set,Wg,Bg)
152     errnew=2*errold
153     while ctt(set,Wg,Bg)>1e-35 and keyboard.is_pressed('q')==False and eps>=1e-100:
154         Wadj,Badj=gradctot(set,Wg,Bg)
155         eps=1
156         while errnew>=errold and eps>=1e-100:
157             W,B,change,leng=adjustment(set, eps,Wadj,Badj)
158             errnew=ctt(set,W,B)
159             eps/=2
160
161         Wg[0:]=W[0:]
162         Bg[0:]=B[0:]
163         errold=errnew
164         errnew=ctt(set,Wg,Bg)
165         print([n,change,float(errnew), leng, eps])
166         H.append([float(errnew),leng, eps])
167
168         if n%1000==0 and n!=0:
169             plt.plot(H[-1000:])
170             plt.legend(['error','leng','eps'])
171             plt.show()
172         n+=1
173     plt.plot(H)
174     plt.legend(['error','leng','eps'])
175     plt.show()
176     return W,B
177 print(train(tset4))
178 print(evalnn(tset4[1][0:3],Wg,Bg))
179 print(evalnn(tset4[2][0:3],Wg,Bg))
180 print(ctt(tset4,Wg,Bg))
```

In der Linien 110 bis 140 wird die Funktion Adjustment programmiert. Mithilfe des Gradientes werden die neue W und B Vektoren berechnet. Die Parameter dieser Funktion sind eps: Epsilon (ein Skalar) und dem Gradienten von ctt als Wadj und Badj. In Linie 111 werden die globalen Variablen für den Weights und Biases importiert, dann werden die drei Nullvektoren bestimmt einer mit 15 Dimensionen, das Zweite mit 5 Dimensionen und das Dritte mit 20 Dimensionen. Die Vektoren des Gradienten (Wadj und Badj) werden danach zusammen in einen einzigen Vektor (das dritte) vereinigt. In Linie 118 wird die Länge des Vektors tot berechnet. Zwischen Zeile 119 und 138 werden den neuen Parametern für W und B berechnet, indem die Differenz zwischen dem gegenwärtigen globalen Parameter und des Gradienten geteilt nach der Länge des Gradienten und multipliziert mit Epsilon. Am Ende der Berechnung des neuen globalen Parametern wird die Durchschnittliche Änderung derselben berechnet. Die Funktion endet mit der Rückgabe der neuen Parameter W und B sowie die durchschnittliche Änderung und die Länge des Gradienten.

9.5 Trainingsschritt mit Kontrolle von ε

Die Letzte Funktion des Training Programm ist die Train Funktion selbst. Diese letzte Funktion hat nur ein Parameter, dem Trainingsset. Erstens werden die Variablen initialisiert: W, B, leng, change, n, H, errold und errnew. Die Globalen variablen Wg und Bg werden einfach importiert.

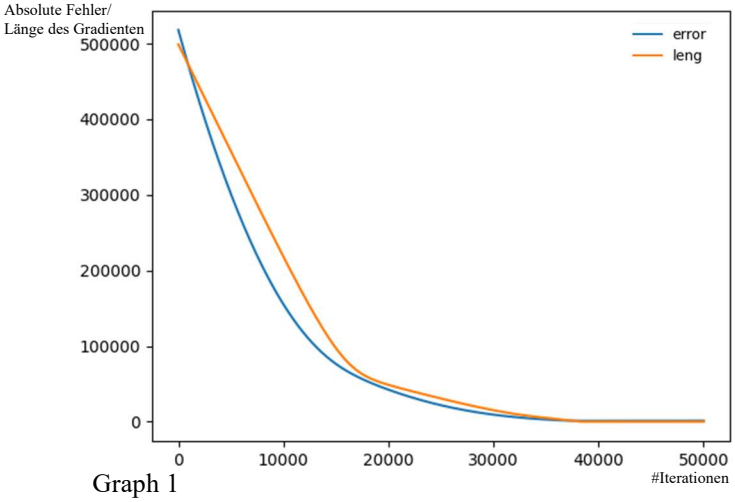
Wenn die Variablen definiert sind, wird das erste while loop programmiert. Solange der allgemeine Fehler grösser als 10^{-35} ist, die Buchstabe q nicht gedruckt ist und eps grösser als10^{-100} ist, dann geht dem loop weiter. Diese Bedingungen sind programmiert damit dem Training ein Ende erreicht bevor all die Ressourcen der Maschine benutzt worden oder wenn merkt man, dass den Modell ein lokales Minimum erreicht hat. In diesem Loop wird erst den Gradienten berechnet und nochmals eps=1 gesetzt.

Epsilon wird zurückgesetzt damit wenn die Schleife mehrmals wiederholt wird, diese Zahl nicht zu klein wird und die Möglichkeit entsteht von einem lokalen Minimum weg zu gehen. In Zeile 156 wird es eine neue while schleife programmiert damit so lange dem neuen Fehler grösser als das alte ist, Epsilon halbiert wird. Die neue Parametern W und B werden neu berechnet. Wenn den Punkt erreicht wird wo dem Fehler sinkt, dann werden die Globale Variablen verändert, somit wird der Fehler immer kleiner. Nach diesem Update werden die Daten bezüglich dieser Verbesserung im Command-line ausgegeben damit es sichtbar wird, was im Programm passiert ist. Im Linie 166 werden diese werten in eine Liste gespeichert. Jede Tausende Iteration wird es eine grafische Darstellung gezeigt wo dem Fehler, die Länge des Gradienten und Epsilon abgebildet sind. Am Ende, wenn die while Bedingungen nicht mehr erfüllt sind, wird ein Graph gezeigt über das ganze Training Prozess. Im Linie 176 werden die trainierte Werte für W und B zurückgegeben und in Linie 177 gedruckt. Nachdem die Funktion beendet wird, werden es zwei Test über das neuronale Netzwerk geführt und dann im Zeile 180 wird der Fehler nochmals berechnet.

10 Das Resultat

Für die Validierung dieses Trainingsmodell sowie für den Versuch dies zu optimieren, wurden verschiedenen Experimenten durchgeführt, welche leicht verschiedenen Modellen als Grundlage haben. Hier folgen einige Resultate zu den verschiedenen Modellen. Ziel diesen Experimenten ist zu überprüfen, ob das NN richtig funktioniert. Alle diesen Modellen benutzen ein, mit zufälligen Werten erstellten Trainingsset. Dieses Set wird erstellt, indem zufällige Werten für W und B gewählt werden. Die erste drei Werten im Trainingspoint sind die zufälligen Werten R, G und B. Die andere zwei werten des Trainingspoint sind dem Output der NN mit den gewählten W und B.

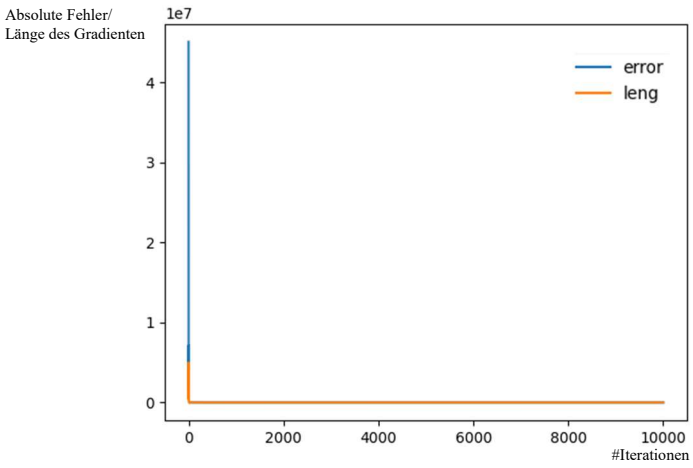
10.1 Versuch der Parameterbestimmung eines NN zur Validierung des Modelles



Graph 1

Dieses Erstes Modell wurde trainiert mit einer Validation Trainingsset, dass keine reellen Bedingungen entspricht, aber dem Modell validieren kann. In der Graph 1 sieht man ein Training, in dem die blaue Linie den Fehler bezeichnet und die orange Linie die Länge des Gradienten bezeichnet. Am Anfang, wenn der

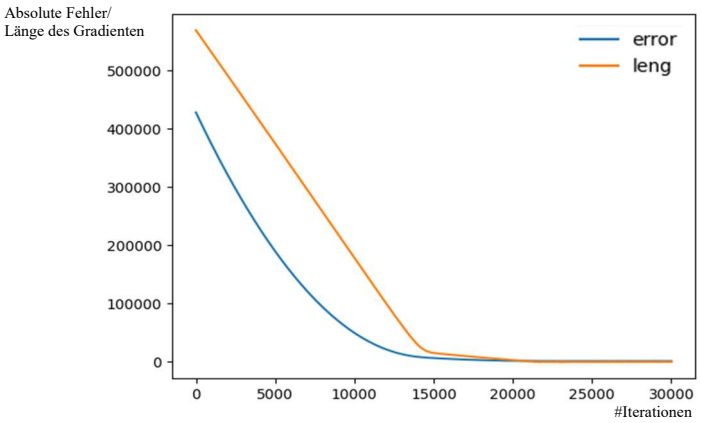
Gradient gross ist, verbessert sich die Fehlerquote schnell, dann je kleiner der Gradient ist, desto langsamer verbessert sich den Fehler. Der Fehler war bei 500’000 dann es hat sich bis auf fast 0 nach 40’000 Iterationen verbessert.



Graph 2

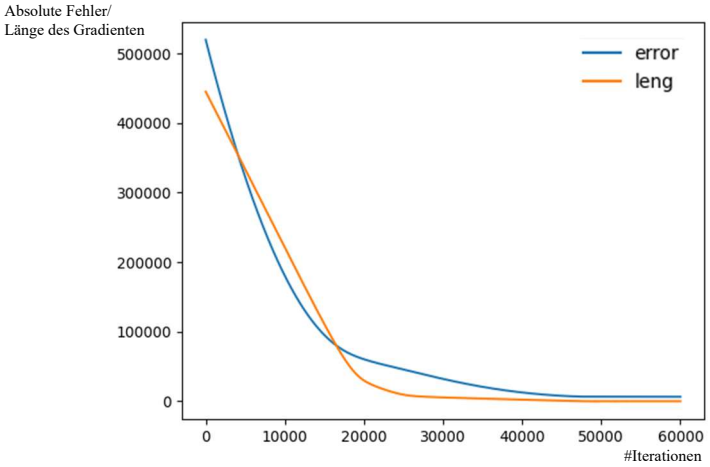
Dieses zweite Modell fängt an mit einem Fehler welche höher als 40'000’000 ist. Nach 1’000 Iterationen hat sich der Wert bis nah am 0 verbessert. Dieser Fehler kann schnell minimiert werden, indem man am Anfang ein grosser Wert für ε wählt. Der Wert wird immer wieder geteilt bis, nah an 0 ist und eine noch bessere Minimierung durchführen kann.

Was anders in all diese Experimenten ist, sind die variablen für W und B welche zufällig initialisiert werden. Eine zweite Bedingung, welche anders ist, ist die Trainingsdauer.



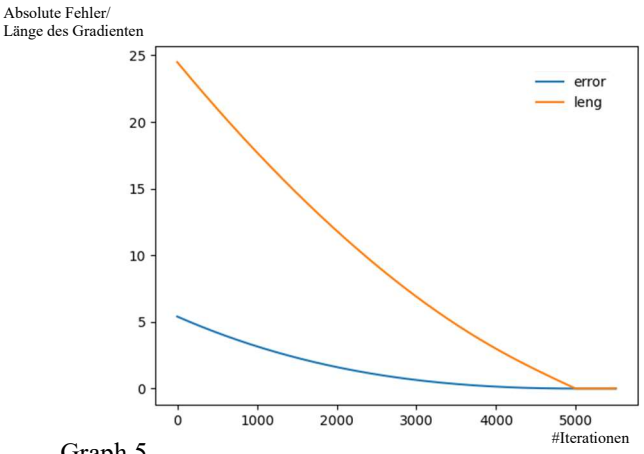
Graph 3

Die Nächsten Zwei Graphen zeigen das Trainingsprozess mit statisches ε. Die zwei Abbildungen wurden mit dem gleichen Modell erstellt, aber mit verschiedenen Anfangswerten für W und B. Den ersten Versuch erreicht ein Fehler von nahezu 0 mit zirka 22’000 Iterationen. Den zweiten Versuch hat nach 60’000 Iterationen immer noch eine höhere Fehlerquote. Dieses Leistungsunterschied kann erklärt werden, mit der Erreichung eines lokalen Minimums.

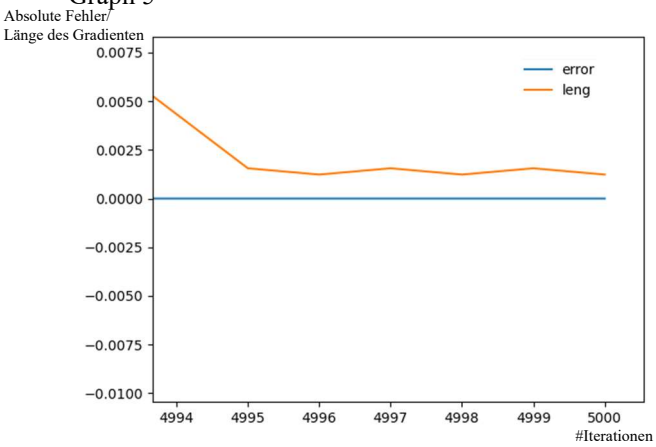


Graph 4

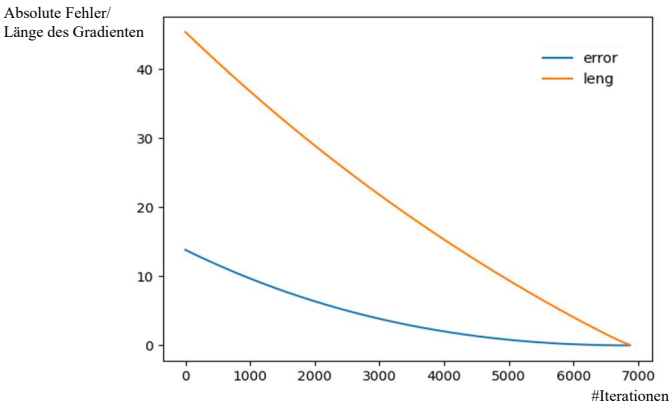
In allen bis jetzt behandelten Graphen war die Aktivierungsfunktion die Identitätsfunktion. Die Änderung der Aktivierungsfunktion kann von dem Fehleranfangswert erkannt werden, welcher bei $f(x) = x$ bis zu 40 Millionen erreichen kann. Im Gegensatz bei der Einsetzung der Sigma Funktion, dem Fehler hat ein Maximum von ~20.



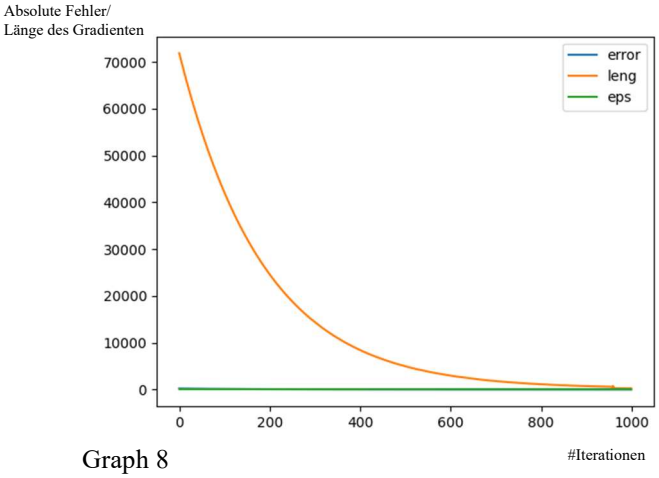
Graph 5



Graph 6



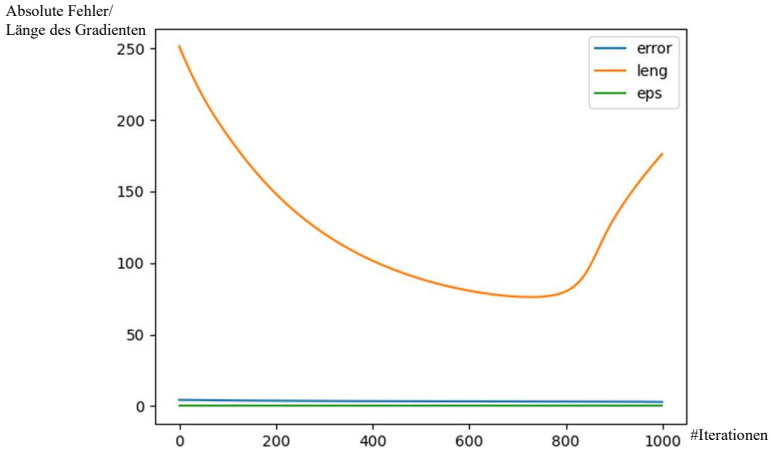
Graph 7



Graph 8

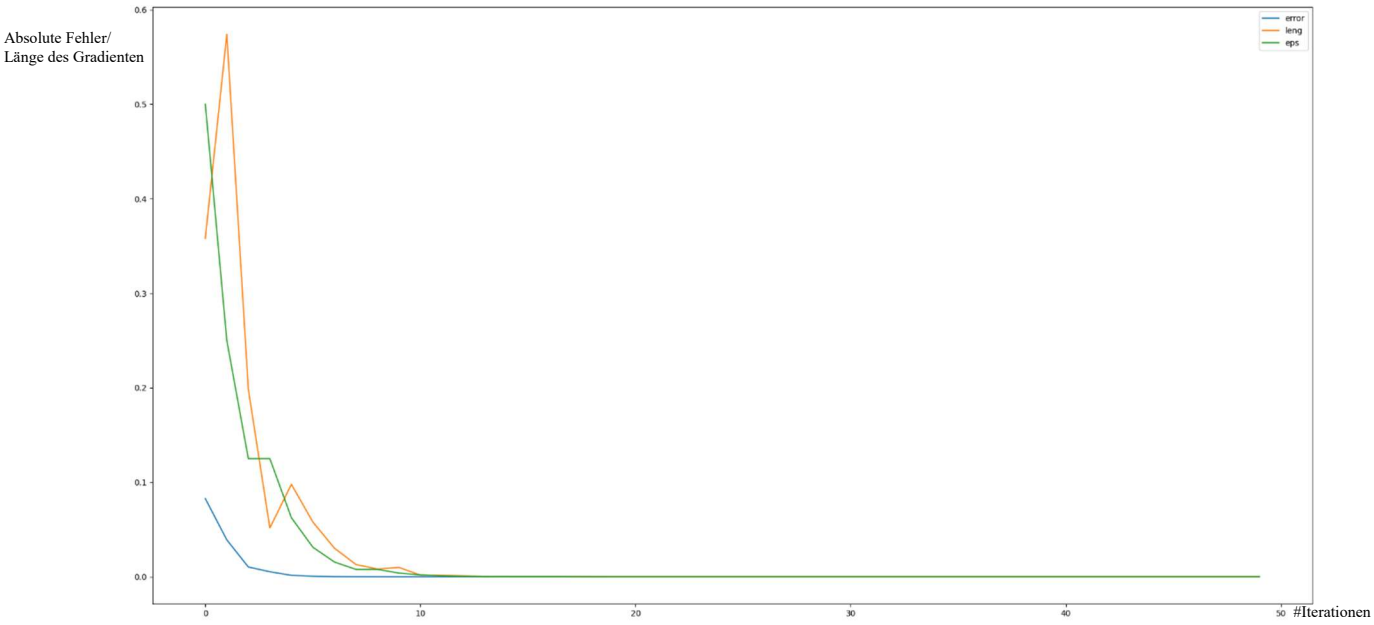
In den nächsten Graphen ist dem Modell anders als in den früheren. Dieses Modell hat eine kürzere Trainingszeit, es braucht weniger als 10'000 Iterationen um dem Minimum zu erreichen und eine höhere Genauigkeit auszuweisen. In den Graphen 5, 6 und 7 sieht man eine quasi-Linearität der Länge des Gradienten und eine sinkende Funktion des Fehlers. Diese zwei Funktionen treffen sich in einem Punkt bei ca. 5'000 bzw bei 6870 Interaktionen. Nach diesem Punkt ist der Fehler praktisch bei 0 und dem Gradienten schwingt zwischen zwei Werten. Dies ist zu sehen im Graph 6. Die beiden Experimente haben das gleiche Algorithmus, aber verschiedenen Anfangswerten für W und B.

Graf 8 zeigt dem Training wo die Aktivierungsfunktion $f_{(x)} = x^3$ benutzt wird. Diese Aktivierungsfunktion, welche unbeschränkt ist, verursacht ein grosser Gradient. Mit dem Laufen des Codes, nähert sich dem Gradienten an 0. Diesen Graphen zeigen, das trotzt die Nutzung eine ungünstige Aktivierungsfunktion, dem Training erfolgreich beendet werden kann. Dies bedeutet aber nicht, dass jede mögliche Funktion als Aktivierungsfunktion



Graph 9

eingesetzt werden kann. Im nächsten Beispiel sieht man warum. Im Graph 9 wird $f_{(x)} = x^2$ als Aktivierungsfunktion benutzt. Diese Funktion bewirkt, dass die Distribution der Outputs nur positiv wird. Dem Fehler kann in diesem Fall nicht minimiert werden. In dem letzten Graph, erfolgt dem Training über viel wenigere Iterationen. Dies erlaubt dem Graphen detaillierter zu sein. In die Abbildung sieht man das alle Werten sinken.



Graph 10

Dem Graph 10 zeigt das Training mit dem Algorithmus welche im Kapitel 9 vorgestellt wird.

11 Diskussion/ Ausblick

In diese Arbeit wurden die Basis Konzepte der KI für die Lösung eines Problems in Kleinformat erklärt. Nach der Veröffentlichung von Chat Gpt, war dem Netz mit Informationen über KI geflutet. Dieser Überschuss an Informationen, die Nicht für dieses Arbeit essenziell waren hat die Selektion der wichtigen Informationen erschwert und dem Aufwand entsprechend erhöht.

Um die Anwendung dieses Wissen auf die Aufgabe mit ein Lego Roboter zu übertragen, wurde schrittweise angegangen.

Das Trainingsprogramm, welche die Variablen W und B bestimmen muss, wurde mit pseudo Trainingssets getestet und zeigt eine ausreichende Genauigkeit. Nach diesem Algorithmus betrug die Kostenfunktion nur noch 9.2×10^{-32} . Dies zeigt das dem Modell funktioniert.

Die Übertragung mit einen Trainingsset welche reellen Bedingungen entspricht konnte nicht durchgeführt werden. Die Schwierigkeit: ein reelles Trainingsset zu erstellen, liegt in der Tatsache, dass die Werte für R, G und B zwar einfach messbar sind, aber die gewünschten Geschwindigkeiten für O0 und O1 nicht eindeutig genug zugeordnet werden können.

Diese Arbeit hat eindeutig gezeigt, dass Die KI vieles kann, aber ein hochqualitatives Trainingsset benötigt wird, damit dem Resultat zufriedenstellend ausfällt. Diese hochqualitativen Trainingssets zu generieren ist eine Zeit und Ressourcenintensive Angelegenheit die zum Teil ins Unendliche führen kann (autonomes Fahren).

12 Abstract

In diese Arbeit wurde es versucht die Frage zu beantworten, ob es möglich ist ein Roboter mit einem neuronalen Netzwerk zu steuern damit dies eine Linie Folgen kann. Um dieses Ziel zu erreichen, wurde es die Theorie umfangreich aufgearbeitet. Diese wurde hier in dem Kapiteln 2, 3 und 4 behandelt. Nachher wurde die Einrichtung der Roboter für die Programmierung in Kapitel 5 erklärt. In den folgenden Kapiteln, 6, 7, 8 und 9 wird das Programm besprochen, welche auf dem Roboter läuft. Danach wird erklärt, wie dem Training Algorithmus funktioniert und welche seine Bestandteile sind. Am Schluss Kapitel 10 wird über verschiedene Graphen diskutiert, welche dem Trainingsprozess durch verschiedene Algorithmen zeigen.

13 Literaturverzeichnis

balena, 2023. *Etcher.* [Online]
Available at: https://etcher.balena.io/
[Zugriff am 7 November 2022].

BBC, 2023. *BBC Future.* [Online]
Available at: https://www.bbc.com/future/article/20190207-technology-in-deep-time-how-it-evolves-alongside-us
[Zugriff am 19 Oktober 2023].

Bloomberg, 2023. *Bloomberg.com.* [Online]
Available at: https://www.bloomberg.com/news/articles/2023-01-23/microsoft-makes-multibillion-dollar-investment-in-openai
[Zugriff am 18 Oktober 2023].

ev3dev, 2023. *Getting Started with ev3dev.* [Online]
Available at: https://www.ev3dev.org/docs/getting-started/#step-1-download-the-latest-ev3dev-image-file
[Zugriff am 10 September 2023].

Ivan Vasilev, D. S. G. S. P. R. V. Y., 2019. *Python Deep Learning Exploring deep learning techniques, neural network architectures and GANs with PyTorch, Keras and TensorFlow.* 2 Hrsg. Birmingham-Mumbai: Packt.

LEGO, 2019. *Lego Education.* [Online]
Available at: https://education.lego.com/en-us/products/lego-mindstorms-education-ev3-core-set/5003400/
[Zugriff am 10 Oktober 2023].

Nielsen, M. A., 2015. *Neural Networks and Deep Learning.* s.l.:Determination Press.

Nilekani, N., 2023. *Medium.com.* [Online]
Available at: https://medium.com/people-and-ai/2023-the-year-of-ai-9ae498ff7593
[Zugriff am 19 Oktober 2023].

Sluka, J., 2009. *A PID Controller For Lego Mindstorms Robots.* [Online]
Available at: https://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html
[Zugriff am 10 September 2023].

The New York Times, 2023. *The New York Times.* [Online]
Available at: https://www.nytimes.com/2023/01/07/technology/generative-ai-chatgpt-investments.html
[Zugriff am 10 Oktober 2023].

14 Anhang

Im Anhang sind alle benutzten Training Algorithmen sowie der Algorithmus, welcher die Trainingssets erstellt und derjenige, welche auf dem Roboter läuft. Leider konnten es keine IDE-ähnliche Ausdrücke erstellt werden. Auf diesen Grund sind die wichtigsten Programme auf GitHub abgespeichert.

https://github.com/gpr05/Maturaarbeit/tree/main

14.1 NN ohne Biases

```
import numpy as np
import matplotlib.pylab as plt

#erste versuch in training
b=np.random.rand(6,1)
a=np.random.rand(9,1)

                        #R                      G               B         re,li
#tset=[[113.11902730375427,153.04650170648463,175.46373720136518,5.0,5.0],[175.0
,175.0,175.0,5.0,2.0],[100.0,100.0,200.0,2.0,5.0]]
#tset2=[[122,200,10,524.3120897791999, 533.6532256],[100, 100, 180,456.66741616,
510.14936],[20, 230, 18,352.413101372, 322.139278],[50, 50, 50,
202.74761647999998, 221.4676],[220, 130, 110, 704.902035172, 781.224002]]

#robset=[[79.56764, 68.08804, 51.28825,-1,1],[116.53203, 124.99206,
78.89265,1,-1],[94.06726, 90.12444, 64.45174,1,1]]

                        #R                      G               B         re,li
tset=[[113.11902730375427,153.04650170648463,175.46373720136518,5.0,5.0],[175.0,
175.0,175.0,5.0,2.0],[100.0,100.0,200.0,2.0,5.0]]
tset2=[[122,200,10,524.3120897791999, 533.6532256],[100, 100, 180,456.66741616,
510.14936],[20, 230, 18,352.413101372, 322.139278],[50, 50, 50,
202.74761647999998, 221.4676],[220, 130, 110, 704.902035172, 781.224002]]

def evalnn(rgb):
    re=(rgb[0]*a[0]*b[0]+rgb[1]*a[1]*b[0]+rgb[2]*a[2]*b[0]+
        rgb[0]*a[3]*b[1]+rgb[1]*a[4]*b[1]+rgb[2]*a[5]*b[1]+
        rgb[0]*a[6]*b[2]+rgb[1]*a[7]*b[2]+rgb[2]*a[8]*b[2])
    li=(rgb[0]*a[0]*b[3]+rgb[1]*a[1]*b[3]+rgb[2]*a[2]*b[3]+
        rgb[0]*a[3]*b[4]+rgb[1]*a[4]*b[4]+rgb[2]*a[5]*b[4]+
        rgb[0]*a[6]*b[5]+rgb[1]*a[7]*b[5]+rgb[2]*a[8]*b[5])
    return re,li
def c(tval):
    re,li=evalnn(tval[0:3])
    return 0.5*(re-tval[3])**2+0.5*(li-tval[4])**2

def ctt(tval):
    ct=0
    for val in tval:
        ct+=c(val)
    return ct

def gradc(tval):
    R=tval[0]
    G=tval[1]
    B=tval[2]
    ret=tval[3]
    lit=tval[4]
    gradb=np.zeros((6,1))
    grada=np.zeros((9,1))
    re,li=evalnn(tval[0:3])
    gradb[0]=(re-ret)*(R*a[0]+G*a[1]+B*a[2])
    gradb[1]=(re-ret)*(R*a[3]+G*a[4]+B*a[5])
    gradb[2]=(re-ret)*(R*a[6]+G*a[7]+B*a[8])
```

```
    gradb[3]=(li-lit)*(R*a[0]+G*a[1]+B*a[2])
    gradb[4]=(li-lit)*(R*a[3]+G*a[4]+B*a[5])
    gradb[5]=(li-lit)*(R*a[6]+G*a[7]+B*a[8])

    grada[0]=(re-ret)*R*b[0]+(li-lit)*R*b[3]
    grada[1]=(re-ret)*G*b[0]+(li-lit)*G*b[3]
    grada[2]=(re-ret)*B*b[0]+(li-lit)*B*b[3]
    grada[3]=(re-ret)*R*b[1]+(li-lit)*R*b[4]
    grada[4]=(re-ret)*G*b[1]+(li-lit)*G*b[4]
    grada[5]=(re-ret)*B*b[1]+(li-lit)*B*b[4]
    grada[6]=(re-ret)*R*b[2]+(li-lit)*R*b[5]
    grada[7]=(re-ret)*G*b[2]+(li-lit)*G*b[5]
    grada[8]=(re-ret)*B*b[2]+(li-lit)*B*b[5]
    return grada, gradb

def gradctot(tval):

    gradbtot=np.zeros((6,1))
    gradatot=np.zeros((9,1))
    for i in tval :#range(3):
        grada, gradb =gradc(i)
        gradatot+=grada
        gradbtot+=gradb

    return gradatot, gradbtot

def adjustment(val,eps):
    c=a
    d=b
    tot=np.zeros((15,1))
    aadj,badj=gradctot(val)
    tot[0:9]=aadj
    tot[8:14]=badj
    lengh=np.linalg.norm(tot)
    a[0]= a[0]-(aadj[0]/lengh)*eps
    a[1]= a[1]-(aadj[1]/lengh)*eps
    a[2]= a[2]-(aadj[2]/lengh)*eps
    a[3]= a[3]-(aadj[3]/lengh)*eps
    a[4]= a[4]-(aadj[4]/lengh)*eps
    a[5]= a[5]-(aadj[5]/lengh)*eps
    a[6]= a[6]-(aadj[6]/lengh)*eps
    a[7]= a[7]-(aadj[7]/lengh)*eps
    a[8]= a[8]-(aadj[8]/lengh)*eps

    b[0]= b[0]-(badj[0]/lengh)*eps
    b[1]= b[1]-(badj[1]/lengh)*eps
    b[2]= b[2]-(badj[2]/lengh)*eps
    b[3]= b[3]-(badj[3]/lengh)*eps
    b[4]= b[4]-(badj[4]/lengh)*eps
    b[5]= b[5]-(badj[5]/lengh)*eps
    middelta=(np.mean(a-c)+np.mean(b-d))/2
    return a,b,middelta, lengh

def train(set,eps):
```

```
    errold=ctt(set)
    errnew=0
    a=None
    b=None
    n=0
    H=[]
    while ctt(set)>0.0 and eps>10**-16:
        if errold-errnew<0:
            eps=eps/2
        a,b,change,leng =adjustment(set,eps)
        errold=errnew
        errnew=ctt(set)
        print([n,change,float(errnew), leng, eps])
        H.append([float(errnew),leng])

        if n%10000==0:
            plt.plot(H)
            plt.show()
        n+=1
    plt.plot(H)
    plt.show()
    return a,b

print(train(tset2,20))
print(evalnn(tset2[0][0:3]))
print(evalnn(tset2[1][0:3]))
print(evalnn(tset2[2][0:3]))
print(ctt(tset2))
```

14.2 NN mit weights und Biases

```
from math import e
import numpy as np
import matplotlib.pylab as plt
import keyboard

#erste versuch in training
W=np.random.rand(18,1)
B=np.random.rand(8,1)

                        #R                       G               B         re,li
tset=[[113.11902730375427,153.04650170648463,175.46373720136518,5.0,5.0],[175.0,
175.0,175.0,5.0,2.0],[100.0,100.0,200.0,2.0,5.0]]
tset2=[[122,200,10,524.3120897791999, 533.6532256],[100, 100, 180,456.66741616,
510.14936],[20, 230, 18,352.413101372, 322.139278],[50, 50, 50,
202.74761647999998, 221.4676],[220, 130, 110, 704.902035172, 781.224002]]


def f(x):
    return 1/(1+e**(-x))

def fder(x):
    return (e**(-x))/((1+e**(-x))**2)

def evalnn(rgb):

re=f(f(f(rgb[0]*W[0]+B[0])*W[3]+f(rgb[1]*W[1]+B[1])*W[4]+f(rgb[2]*W[2]+B[2])*W[5
]+B[3])*W[12]+

f(f(rgb[0]*W[0]+B[0])*W[6]+f(rgb[1]*W[1]+B[1])*W[7]+f(rgb[2]*W[2]+B[2])*W[8]+B[4
])*W[13]+

f(f(rgb[0]*W[0]+B[0])*W[9]+f(rgb[1]*W[1]+B[1])*W[10]+f(rgb[2]*W[2]+B[2])*W[11]+B
[5])*W[14]+B[6])

li=f(f(f(rgb[0]*W[0]+B[0])*W[3]+f(rgb[1]*W[1]+B[1])*W[4]+f(rgb[2]*W[2]+B[2])*W[5
]+B[3])*W[15]+

f(f(rgb[0]*W[0]+B[0])*W[6]+f(rgb[1]*W[1]+B[1])*W[7]+f(rgb[2]*W[2]+B[2])*W[8]+B[4
])*W[16]+

f(f(rgb[0]*W[0]+B[0])*W[9]+f(rgb[1]*W[1]+B[1])*W[10]+f(rgb[2]*W[2]+B[2])*W[11]+B
[5])*W[17]+B[7])
    return re,li

def c(tval):
    re,li=evalnn(tval[0:3])
    return 0.5*(re-tval[3])**2+0.5*(li-tval[4])**2

def ctt(tval):
    ct=0
    for val in tval:
        ct+=c(val)
    return ct
```

```
def gradc(tval):
    R=tval[0]
    G=tval[1]
    b=tval[2]
    ret=tval[3]
    lit=tval[4]
    gradW=np.zeros((18,1))
    gradB=np.zeros((8,1))
    re,li=evalnn(tval[0:3])
    gradW[0]=
(re-ret)*fder(f(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3]
)*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  R+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  R+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*R+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  R+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  R+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*R+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))


gradW[1]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*
W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(
```

```
(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* G+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* G+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*G+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* G+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* G+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*G+fder(b*W[2]+B[2])*W[11]*0)))


gradW[2]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*
W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* b))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* b))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*b)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(
```

```
(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* b))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* b))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*b)))

    gradW[3]=
(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3]
)*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  1+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  1+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))

    gradW[4]=
(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3]
)*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(
```

```
(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 1+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 1+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))

    gradW[5]=
(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3]
)*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 1))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(
```

```
(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 1))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))

    gradW[6]=
(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3]
)*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  1+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  1+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))

    gradW[7]=
(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3]
)*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(
```

```
(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 1+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 1+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))

    gradW[8]=
(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3]
)*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 1))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(
```

```
(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 1))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))

    gradW[9]=
(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3]
)*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*1+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*1+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))


gradW[10]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])
*W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(
```

```
(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*1+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*1+fder(b*W[2]+B[2])*W[11]*0)))


gradW[11]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])
*W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*1)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(
```

```
(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*1)))


gradW[12]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])
*W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(


(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*1)+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*0*(fder(
R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*0*(fde
r(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))


gradW[13]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])
*W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*0*(fder(
R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*1)+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*0*(fde
r(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))


gradW[14]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])
*W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*0*(fder(
```

```
R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*0*(fder(
R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*1))


gradW[15]=(li-lit)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])
*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*1)+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*0*(fder(
R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*0*(fde
r(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))


gradW[16]=(li-lit)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])
*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*0*(fder(
R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*1)+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*0*(fde
r(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))


gradW[17]=(li-lit)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])
*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*0*(fder(
R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*0*(fder(
```

```
R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*1))


gradB[0]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*
W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  1+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  1+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*1+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  1+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  1+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*1+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)))


gradB[1]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*
W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 1+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 1+fder(b*W[2]+B[2])*W[8]* 0))+
```

```
(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*1+fder(b*W[2]+B[2])*W[11]*0)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 1+fder(b*W[2]+B[2])*W[5]* 0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 1+fder(b*W[2]+B[2])*W[8]* 0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*1+fder(b*W[2]+B[2])*W[11]*0)))


gradB[2]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*
W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 1))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 1))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*1)))
+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*(f
der(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]* 1))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*(f
der(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]* 1))+
```

```
(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*1)))


gradB[3]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*
W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*1)
+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*0*
(fder(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]*
0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
0*(fder(R*W[0]+B[0])*W[9]*1+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)
))+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*1)
+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*0*
(fder(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]*
0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
0*(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)
))


gradB[4]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*
W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*0*
(fder(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]*
```

```
0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*1)
+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
0*(fder(R*W[0]+B[0])*W[9]*0+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)
))+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*0*
(fder(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]*
0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*1)
+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
0*(fder(R*W[0]+B[0])*W[9]*1+fder(G*W[1]+B[1])*W[10]*0+fder(b*W[2]+B[2])*W[11]*0)
))


gradB[5]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*
W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*(

(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[12]*0*
(fder(R*W[0]+B[0])*W[3]*0+fder(G*W[1]+B[1])*W[4]*0+fder(b*W[2]+B[2])*W[5]*0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]*0*
(fder(R*W[0]+B[0])*W[6]*0+fder(G*W[1]+B[1])*W[7]*0+fder(b*W[2]+B[2])*W[8]*0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]*
1))+(li-lit)*fder(


f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*(
```

```
(fder(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]*0*
(fder(R*W[0]+B[0])*W[3]*  0+fder(G*W[1]+B[1])*W[4]* 0+fder(b*W[2]+B[2])*W[5]*
0))+

(fder(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]*0*
(fder(R*W[0]+B[0])*W[6]*  0+fder(G*W[1]+B[1])*W[7]* 0+fder(b*W[2]+B[2])*W[8]*
0))+

(fder(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]*
1))


gradB[6]=(re-ret)*fder(f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*
W[5]+B[3])*W[12]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[13]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[14]+B[6]
)*1

    gradB[7]=(li-lit)*fder(
f(f(R*W[0]+B[0])*W[3]+f(G*W[1]+B[1])*W[4]+f(b*W[2]+B[2])*W[5]+B[3])*W[15]+

f(f(R*W[0]+B[0])*W[6]+f(G*W[1]+B[1])*W[7]+f(b*W[2]+B[2])*W[8]+B[4])*W[16]+

f(f(R*W[0]+B[0])*W[9]+f(G*W[1]+B[1])*W[10]+f(b*W[2]+B[2])*W[11]+B[5])*W[17]+B[7]
)*1
    return gradW, gradB

def gradctot(tval):

    gradWtot=np.zeros((18,1))
    gradBtot=np.zeros((8,1))
    for i in tval :
        gradW, gradB =gradc(i)
        gradWtot+=gradW
        gradBtot+=gradB
    return gradWtot, gradBtot

def adjustment(val,eps):
    c=W
    d=B
    tot=np.zeros((26,1))
    Wadj,Badj=gradctot(val)
    tot[0:18]=Wadj
    tot[17:25]=Badj
    lengh=np.linalg.norm(tot)
    W[0]=W[0]-(Wadj[0]/lengh)*eps
    W[1]=W[1]-(Wadj[1]/lengh)*eps
    W[2]=W[2]-(Wadj[2]/lengh)*eps
    W[3]=W[3]-(Wadj[3]/lengh)*eps
    W[4]=  W[4]-(Wadj[4]/lengh)*eps
    W[5]=  W[5]-(Wadj[5]/lengh)*eps
    W[6]=  W[6]-(Wadj[6]/lengh)*eps
```

```
    W[7]=  W[7]-(Wadj[7]/lengh)*eps
    W[8]=  W[8]-(Wadj[8]/lengh)*eps
    W[9]=  W[9]-(Wadj[9]/lengh)*eps
    W[10]=W[10]-(Wadj[10]/lengh)*eps
    W[11]=W[11]-(Wadj[11]/lengh)*eps
    W[12]=W[12]-(Wadj[12]/lengh)*eps
    W[13]=W[13]-(Wadj[13]/lengh)*eps
    W[14]=W[14]-(Wadj[14]/lengh)*eps
    W[15]=W[15]-(Wadj[15]/lengh)*eps
    W[16]=W[16]-(Wadj[16]/lengh)*eps
    W[17]=W[17]-(Wadj[17]/lengh)*eps
    B[0]=  B[0]-(Badj[0]/lengh)*eps
    B[1]=  B[1]-(Badj[2]/lengh)*eps
    B[2]=  B[2]-(Badj[2]/lengh)*eps
    B[3]=  B[3]-(Badj[3]/lengh)*eps
    B[4]=  B[4]-(Badj[4]/lengh)*eps
    B[5]=  B[5]-(Badj[5]/lengh)*eps
    B[6]=  B[6]-(Badj[6]/lengh)*eps
    B[7]=  B[7]-(Badj[7]/lengh)*eps
    middelta=(np.mean(W-c)+np.mean(B-d))/2
    return W,B,middelta, lengh

def train(set,eps):
    errold=ctt(set)
    errnew=0
    W=None
    B=None
    n=0
    H=[]
    while ctt(set)>0.0 and keyboard.is_pressed('q')==False:
        if errold-errnew<0:
            eps/=1
        elif errold==errnew:
            eps*=1
        W,B,change,leng =adjustment(set,eps)
        errold=errnew
        errnew=ctt(set)
        print([n,change,float(errnew), leng, eps])
        H.append([float(errnew),leng])

        if n%5000==0:
            plt.plot(H)
            plt.show()
        n+=1
    plt.plot(H)
    plt.show()
    return W,B
print(train(tset2,0.0001))
print(evalnn(tset2[1][0:3]))
print(evalnn(tset2[2][0:3]))
print(ctt(tset2))
```

14.3 Modulare NN

```
from math import exp
import numpy as np
import matplotlib.pylab as plt
import keyboard

Wg=np.random.rand(15,1)
Bg=np.random.rand(5,1)

tset4=[[46.14942747766261, 247.36208389705283, 74.08030744641174, 0.81659393,
0.91809978],[133.33315332807823, 117.71784034715313, 108.35543734187672,
0.81659393, 0.91809978],
       [197.66346409625467, 183.56815506745406, 134.3071830935232, 0.81659393,
0.91809978],[189.02759701085495, 82.06516146029526, 49.941384305976115,
0.81659393, 0.91809978],
       [207.77045285493006, 28.652714608810204, 117.00318360637586, 0.81659393,
0.91809978],[217.06198467287302, 254.42309173931255, 113.27118072812857,
0.81659393, 0.91809978],
       [200.69802924376225, 150.14609728332036, 24.68228158639408, 0.81659393,
0.91809978],[104.50758209407022, 213.72096341760204, 80.64609595214219,
0.81659393, 0.91809978],
       [234.60614035888858, 92.50591927648051, 169.84191157618739, 0.81659393,
0.91809978],[192.15905854374887, 23.788168811017936, 152.64503882316762,
0.81659393, 0.91809978],
       [135.02355074609744, 115.45833034239008, 125.95283619782644, 0.81659393,
0.91809978],[215.5057541073719, 221.50080521570374, 83.85481371566169,
0.81659393, 0.91809978],
       [183.52959442689735, 200.9138247697748, 135.5292693627817, 0.81659393,
0.91809978],[195.34023292787631, 123.34621893684788, 107.18389635903542,
0.81659393, 0.91809978],
       [179.6591004825346, 201.39125956349358, 241.52247183974075, 0.81659393,
0.91809978]]

def f(x):
    return 1/(1+exp(-x))

def fder(x):
    return (exp(-x))/((1+exp(-x))**2)
def H0pre(inp,W,B):
    return inp[0]*W[0]+inp[1]*W[1]+inp[2]*W[2]+B[0]

def H0(inp,W,B):
    return f(H0pre(inp,W,B))

def H1pre(inp,W,B):
    return inp[0]*W[3]+inp[1]*W[4]+inp[2]*W[5]+B[1]

def H1(inp,W,B):
    return f(H1pre(inp,W,B))

def H2pre(inp,W,B):
    return inp[0]*W[6]+inp[1]*W[7]+inp[2]*W[8]+B[2]

def H2(inp,W,B):
    return f(H2pre(inp,W,B))
```

```
def O0pre(inp,W,B):
    return
float(H0(inp,W,B))*W[9]+float(H1(inp,W,B))*W[10]+float(H2(inp,W,B))*W[11]+B[3]

def O0(inp,W,B):
    return f(O0pre(inp,W,B))

def O1pre(inp,W,B):
    return
float(H0(inp,W,B))*W[12]+float(H1(inp,W,B))*W[13]+float(H2(inp,W,B))*W[14]+B[4]

def O1(inp,W,B):
    return f(O1pre(inp,W,B))

def evalnn(inp,W,B):
    re=O0(inp,W,B)
    li=O1(inp,W,B)
    return re,li

def c(tval,W,B):
    re,li=evalnn(tval[0:3],W,B)
    return 0.5*(re-tval[3])**2+0.5*(li-tval[4])**2

def ctt(tval,W,B):
    ct=0
    for val in tval:
        ct+=c(val,W,B)
    return ct

def gradc(tval,W,B):
    inp=[0.0,0.0,0.0]
    inp[0:3]=tval[0:3]
    ret=tval[3]
    lit=tval[4]
    gradW=np.zeros((15,1))
    gradB=np.zeros((5,1))
    re,li=evalnn(tval[0:3],W,B)

    gradW[0]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[0]*W[9]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[0]*W[12]
    gradW[1]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[1]*W[9]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[1]*W[12]
    gradW[2]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[2]*W[9]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*inp[2]*W[12]
    gradW[3]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[0]*W[10]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[0]*W[13]
    gradW[4]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[1]*W[10]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[1]*W[13]
```

```
    gradW[5]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[2]*W[10]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*inp[2]*W[13]
    gradW[6]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[0]*W[11]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[0]*W[14]
    gradW[7]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[1]*W[11]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[1]*W[14]
    gradW[8]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[2]*W[11]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*inp[2]*W[14]

    gradW[9]= (re-ret)*float(fder(O0pre(inp,W,B)))*float(H0(inp,W,B))
    gradW[10]=(re-ret)*float(fder(O0pre(inp,W,B)))*float(H1(inp,W,B))
    gradW[11]=(re-ret)*float(fder(O0pre(inp,W,B)))*float(H2(inp,W,B))
    gradW[12]=(li-lit)*float(fder(O1pre(inp,W,B)))*float(H0(inp,W,B))
    gradW[13]=(li-lit)*float(fder(O1pre(inp,W,B)))*float(H1(inp,W,B))
    gradW[14]=(li-lit)*float(fder(O1pre(inp,W,B)))*float(H2(inp,W,B))

    gradB[0]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*W[9]
+(li-lit)*float(fder(O1pre(inp,W,B)))*float(fder(H0pre(inp,W,B)))*W[12]
    gradB[1]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*W[10]+(li-lit)*
float(fder(O1pre(inp,W,B)))*float(fder(H1pre(inp,W,B)))*W[13]
    gradB[2]=
(re-ret)*float(fder(O0pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*W[11]+(li-lit)*
float(fder(O1pre(inp,W,B)))*float(fder(H2pre(inp,W,B)))*W[14]
    gradB[3]= (re-ret)*float(fder(O0pre(inp,W,B)))
    gradB[4]= (li-lit)*float(fder(O1pre(inp,W,B)))
    return gradW, gradB

def gradctot(tval,W,B):
    gradWtot=np.zeros((15,1))
    gradBtot=np.zeros((5,1))
    for i in tval :
        gradW, gradB =gradc(i,W,B)
        gradWtot+=gradW
        gradBtot+=gradB
    return gradWtot, gradBtot

def adjustment(eps,Wadj,Badj):
    global Wg,Bg
    W=np.zeros((15,1))
    B=np.zeros((5,1))
    tot=np.zeros((20,1))

    tot[0:15]=Wadj
    tot[15:20]=Badj
    lengh=np.linalg.norm(tot)
    W[0]=Wg[0]-(Wadj[0]/lengh)*eps
    W[1]=Wg[1]-(Wadj[1]/lengh)*eps
    W[2]=Wg[2]-(Wadj[2]/lengh)*eps
```

```
    W[3]=Wg[3]-(Wadj[3]/lengh)*eps
    W[4]=  Wg[4]-(Wadj[4]/lengh)*eps
    W[5]=  Wg[5]-(Wadj[5]/lengh)*eps
    W[6]=  Wg[6]-(Wadj[6]/lengh)*eps
    W[7]=  Wg[7]-(Wadj[7]/lengh)*eps
    W[8]=  Wg[8]-(Wadj[8]/lengh)*eps
    W[9]=  Wg[9]-(Wadj[9]/lengh)*eps
    W[10]=Wg[10]-(Wadj[10]/lengh)*eps
    W[11]=Wg[11]-(Wadj[11]/lengh)*eps
    W[12]=Wg[12]-(Wadj[12]/lengh)*eps
    W[13]=Wg[13]-(Wadj[13]/lengh)*eps
    W[14]=Wg[14]-(Wadj[14]/lengh)*eps
    B[0]=  Bg[0]-(Badj[0]/lengh)*eps
    B[1]=  Bg[1]-(Badj[2]/lengh)*eps
    B[2]=  Bg[2]-(Badj[2]/lengh)*eps
    B[3]=  Bg[3]-(Badj[3]/lengh)*eps
    B[4]=  Bg[4]-(Badj[4]/lengh)*eps
    middelta=(np.mean(Wg-W)+np.mean(Bg-B))/2
    return W,B, middelta, lengh

def train(set):
    global Wg,Bg
    W=np.zeros((15,1))
    B=np.zeros((5,1))
    leng=None
    change=None
    n=0
    eps=1
    H=[]
    errold=ctt(set,Wg,Bg)
    errnew=2*errold
    while ctt(set,Wg,Bg)>1e-50 and keyboard.is_pressed('q')==False and
eps>=1e-300:
        Wadj,Badj=gradctot(set,Wg,Bg)
        eps=1
        while errnew>=errold and eps>=1e-300:
            W,B,change,leng=adjustment(eps,Wadj,Badj)
            errnew=ctt(set,W,B)
            eps/=2

        Wg[0:]=W[0:]
        Bg[0:]=B[0:]
        errold=errnew
        errnew=ctt(set,Wg,Bg)
        print([n,change,float(errnew), leng, eps])
        H.append([float(errnew),leng, eps])

        if n%1000==0 and n!=0:
            plt.plot(H[-1000:])
            plt.legend(['error','leng','eps'])
            plt.show()
        n+=1
    plt.plot(H)
    plt.legend(['error','leng','eps'])
```

```
    plt.show()
    return W,B
print(train(tset4))
print(evalnn(tset4[1][0:3],Wg,Bg))
print(evalnn(tset4[2][0:3],Wg,Bg))
print(ctt(tset4,Wg,Bg))
```

14.4 Trainingset Generator

```
import numpy as np
import matplotlib.pylab as plt
from math import e
import pandas as pd
import os

b=np.array([0.95764,0.45682,0.52884,0.4,0.566,1.23])
a=np.array([1.09934,0.9454,0.345,0.8993,0.3452,0.0345,1.236,0.4444,0.555])
rgb=[]
B=np.random.rand(5,1)#(8,1)
W=np.random.rand(15,1)#(18,1)
print(B,W)

def evalnn(rgb):
    re=(rgb[0]*a[0]*b[0]+rgb[1]*a[1]*b[0]+rgb[2]*a[2]*b[0]+
        rgb[0]*a[3]*b[1]+rgb[1]*a[4]*b[1]+rgb[2]*a[5]*b[1]+
        rgb[0]*a[6]*b[2]+rgb[1]*a[7]*b[2]+rgb[2]*a[8]*b[2])
    li=(rgb[0]*a[0]*b[3]+rgb[1]*a[1]*b[3]+rgb[2]*a[2]*b[3]+
        rgb[0]*a[3]*b[4]+rgb[1]*a[4]*b[4]+rgb[2]*a[5]*b[4]+
        rgb[0]*a[6]*b[5]+rgb[1]*a[7]*b[5]+rgb[2]*a[8]*b[5])
    return re,li



def f(x):
    return 1/(1+e**(-x))#x#1/(1+e**(-10*x+5))

def fder(x):
    return (e**(-x))/((1+e**(-x))**2)#1#(e**10*x+5)/((1+e**10*x+5)**2)

def evalnnWB(rgb):

re=f(f(f(rgb[0]*W[0]+B[0])*W[3]+f(rgb[1]*W[1]+B[1])*W[4]+f(rgb[2]*W[2]+B[2])*W[5
]+B[3])*W[12]+

f(f(rgb[0]*W[0]+B[0])*W[6]+f(rgb[1]*W[1]+B[1])*W[7]+f(rgb[2]*W[2]+B[2])*W[8]+B[4
])*W[13]+

f(f(rgb[0]*W[0]+B[0])*W[9]+f(rgb[1]*W[1]+B[1])*W[10]+f(rgb[2]*W[2]+B[2])*W[11]+B
[5])*W[14]+B[6])

li=f(f(f(rgb[0]*W[0]+B[0])*W[3]+f(rgb[1]*W[1]+B[1])*W[4]+f(rgb[2]*W[2]+B[2])*W[5
]+B[3])*W[15]+

f(f(rgb[0]*W[0]+B[0])*W[6]+f(rgb[1]*W[1]+B[1])*W[7]+f(rgb[2]*W[2]+B[2])*W[8]+B[4
])*W[16]+

f(f(rgb[0]*W[0]+B[0])*W[9]+f(rgb[1]*W[1]+B[1])*W[10]+f(rgb[2]*W[2]+B[2])*W[11]+B
[5])*W[17]+B[7])
    return rgb,re,li

def H0pre(inp):
    return inp[0]*W[0]+inp[1]*W[1]+inp[2]*W[2]+B[0]
```

```
def H0(inp):
    return f(H0pre(inp))

def H1pre(inp):
    return inp[0]*W[3]+inp[1]*W[4]+inp[2]*W[5]+B[1]

def H1(inp):
    return f(H1pre(inp))

def H2pre(inp):
    return inp[0]*W[6]+inp[1]*W[7]+inp[2]*W[8]+B[2]

def H2(inp):
    return f(H2pre(inp))

def O0pre(inp):
    return H0(inp)*W[9]+H1(inp)*W[10]+H2(inp)*W[11]+B[3]

def O0(inp):
    return f(O0pre(inp))

def O1pre(inp):
    return H0(inp)*W[12]+H1(inp)*W[13]+H2(inp)*W[14]+B[4]

def O1(inp):
    return f(O1pre(inp))

def evalnn3(inp):
    re=O0(inp)
    li=O1(inp)
    return inp,re,li

def eval(k):
    for i in range(k):
        d=[0.0,0.0,0.0]
        for r in range(3):
            d[r]=np.random.random()*255
        rgb.append([evalnn3(d)])
    for g in range(k):
        print(rgb[g])
eval(25)
```

14.5 NN auf dem Roboter

```
#!/usr/bin/env python3
from math import exp
from ev3dev2 import*
from ev3dev2.motor import LargeMotor,OUTPUT_A,OUTPUT_B, SpeedPercent
from ev3dev2.sensor import INPUT_2
from ev3dev2.sensor.lego import ColorSensor

#TODO:
rm= LargeMotor(OUTPUT_A)
lm= LargeMotor(OUTPUT_B)
lf=ColorSensor(INPUT_2)

W=[
0.22761073,0.31168518,0.98887353,0.41346739,0.28467911,0.76328284,0.62464784,0.4
4168643,
        0.17191695,0.01403663,-0.40239056,-0.12775812,-0.6416017
,-0.23089563,0.13047133]
B=[ 0.99682834,0.79983418,0.37870556,-0.17703514,0.04887882]

def f(x):
    return 1/(1+exp(-x))

def H0pre(inp,W,B):
    return inp[0]*W[0]+inp[1]*W[1]+inp[2]*W[2]+B[0]

def H0(inp,W,B):
    return f(H0pre(inp,W,B))

def H1pre(inp,W,B):
    return inp[0]*W[3]+inp[1]*W[4]+inp[2]*W[5]+B[1]

def H1(inp,W,B):
    return f(H1pre(inp,W,B))

def H2pre(inp,W,B):
    return inp[0]*W[6]+inp[1]*W[7]+inp[2]*W[8]+B[2]

def H2(inp,W,B):
    return f(H2pre(inp,W,B))

def O0pre(inp,W,B):
    return
float(H0(inp,W,B))*W[9]+float(H1(inp,W,B))*W[10]+float(H2(inp,W,B))*W[11]+B[3]

def O0(inp,W,B):
    return f(O0pre(inp,W,B))

def O1pre(inp,W,B):
    return
float(H0(inp,W,B))*W[12]+float(H1(inp,W,B))*W[13]+float(H2(inp,W,B))*W[14]+B[4]

def O1(inp,W,B):
    return f(O1pre(inp,W,B))
```

```
def evalnn(inp,W,B):
    re=O0(inp,W,B)
    li=O1(inp,W,B)
    return re,li
while True:
    inp=lf.rgb
    r,l=evalnn(inp,W,B)
    rm.on(SpeedPercent(r*10))
    lm.on(SpeedPercent(l*10))
```

15 Eigenständigkeitserklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit ohne fremde Hilfe und ohne Verwendung anderer als der angegebenen Hilfsmittel verfasst habe,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss der gängigen wissenschaftlichen Zitierregeln korrekt zitiert habe.

Ort / Datum Prada, den 20.10.2023 Unterschrift: