Fraunhofer IAIS

KI und Programmierung

Künstliche Neuronale Netze mit dem Open Roberta Lab

Thorsten Leimbach, Geschäftsfeld Smart Coding and Learning

Version: 2.0

ROBERTA INITIATIVE

OPEN ROBERTA

Neuronale Netze

Erläuterung

Diese Folien sollen einen Überblick über die Grundfunktionalitäten der Integration von Künstliche Neuronale Netzen im Open Roberta Lab geben. Es wird erläutert, wie ein Künstliche Neuronale Netz im Open Roberta Lab selbst erstellt und in einem eigenen NEPO (Open Roberta) Programm verwendet werden kann.

Diese Folien sollen dazu dienen, die grundlegende Handhabung Neuronaler Netze mit Open Roberta zu verstehen.

Übersicht

1. Neuronale Netze mit dem Open Roberta Lab
2. Beispiele für die Berechnung eines Neuronalen Netzes
3. Einbetten eines Netzes in einem Programm
4. (LERNEN mit xNN)
5. Ausblick

Open Roberta xNN wird vom Ministerium für Schule und Bildung des Landes NRW gefördert und in Zusammenarbeit mit Lehrkräften entwickelt:
https://www.iais.fraunhofer.de/de/presse/presseinformationen/presseinformationen-2022/presseinformation-22mmtt2.html

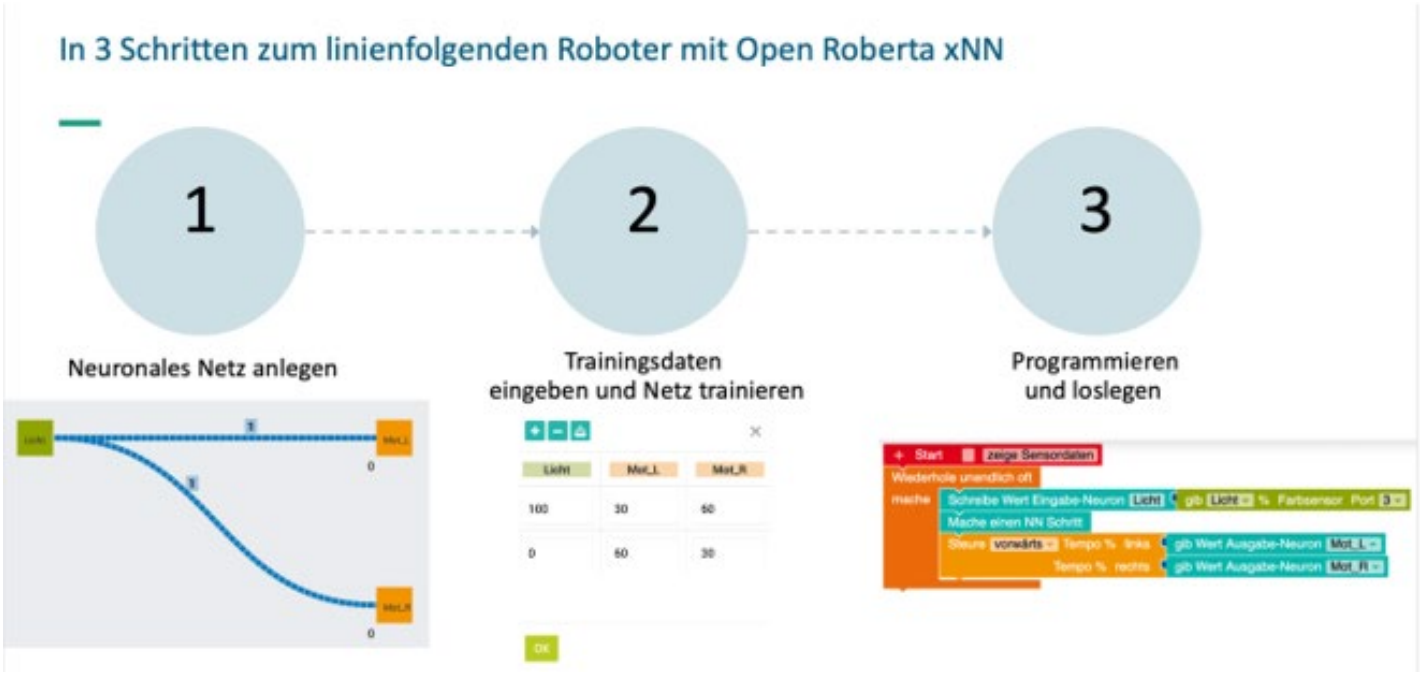
Neuronale Netze

Wie funktioniert ein Neuronales Netz im Open Roberta Lab?

Seit 2022 ist es möglich, im Open Roberta Lab Künstliche Neuronale Netze zu programmieren. Die Optionen bei »Open Roberta xNN« umfassen:

»NEURONALES NETZ Definieren«, hier kann das Neuronale Netz festgelegt werden. (1)

- »NEURONALES NETZ Lernen«, in dieser kann Lernen aus Trainingsdaten durchgeführt werden. (2)
- In der Ansicht "Programm" gibt es Blöcke, mit denen Daten mit dem Neuronalen Netzes ausgetauscht und berechnet werden können. (3)



Ziel der Integration von Künstlichen Neuronalen Netzen (KNN) in Open Roberta Lab ist es, die Grundlagen von KNN zu vermitteln und zu zeigen, wie diese anhand eines Roboters eingesetzt werden können. Die Integration von KNN wird Open Roberta xNN genannt. xNN steht für explainable Neural Networks.

Kapitel 01

Programmieren mit dem Open Roberta Lab

Fraunhofer IAIS

Open Roberta Lab kennenlernen

Einsteiger-Tutorial für Neulinge

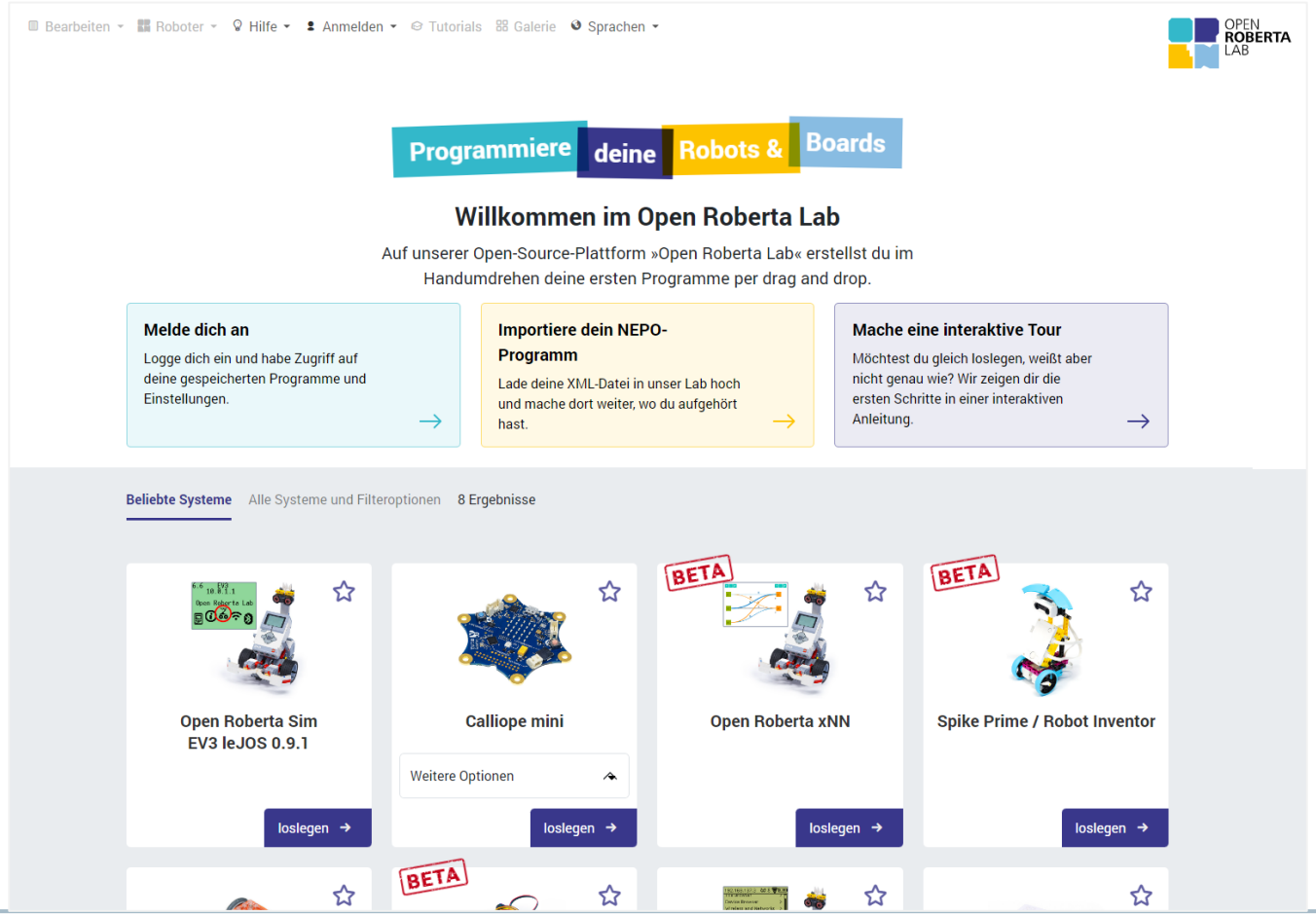
Für Einsteiger*innen wird empfohlen, die »Tour« aufzurufen.

Hierzu:

1. Open Roberta Lab: https://lab.open-roberta.org
2. Klick auf Starte die Tour, um ein interaktives Tutorial zu starten. Es zeigt die prinzipielle Handhabung des Open Roberta Labs und die Nutzung der Simulation.

Weitere Informationen gibt es im offiziellen **Open Roberta Wiki** und auf der Projektwebseite:

- https://wiki.open-roberta.org
- www.open-roberta.org



Fraunhofer IAIS

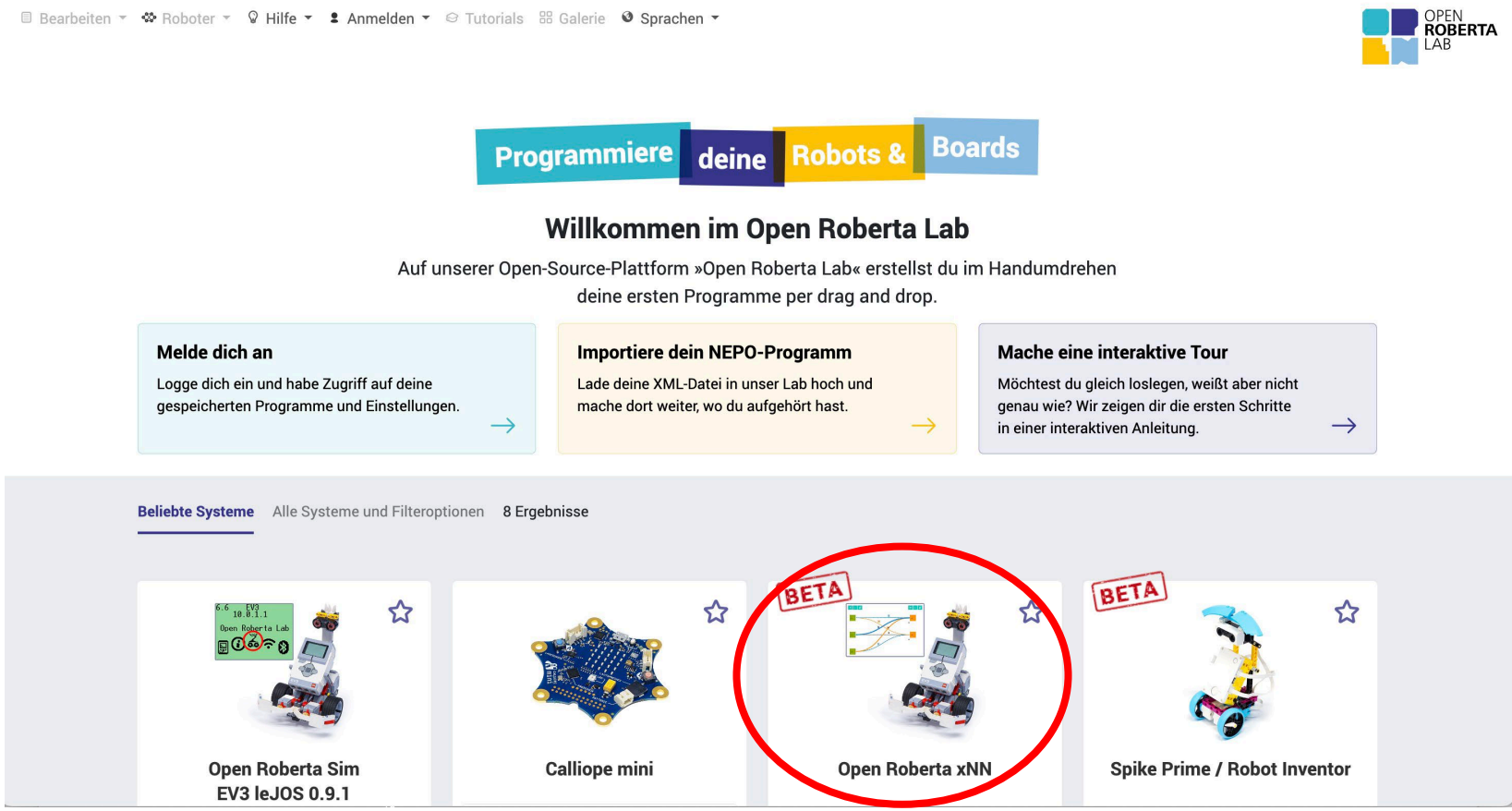
Open Roberta Lab

Neuronale Netze programmieren mit Open Roberta xNN

Um die Programmierung Künstlicher Neuronaler Netze zu starten, muss in der Auswahl der vorhandenen Systeme **Open Roberta xNN** ausgewählt werden

Alternativ kann **Open Roberta xNN** direkt über folgenden Link aufgerufen werden:

- https://lab.open-roberta.org/?loadSystem=xNN

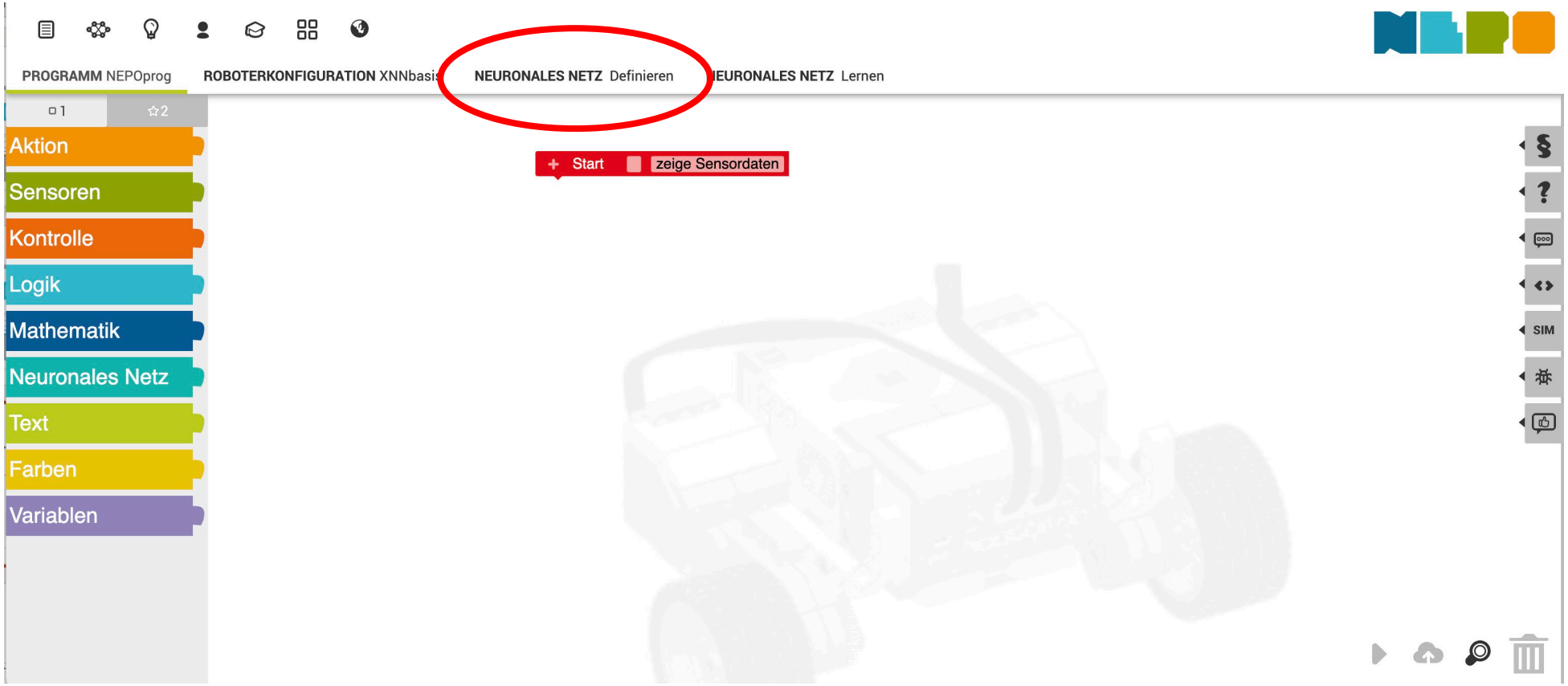


Fraunhofer IAIS

Open Roberta Lab

Neuronale Netze programmieren mit Open Roberta xNN

Dies ist die Standardansicht des Open Roberta Lab. Im Tab **NEURONALES NETZ Definieren** wird das Neuronale Netz angezeigt und konfiguriert.

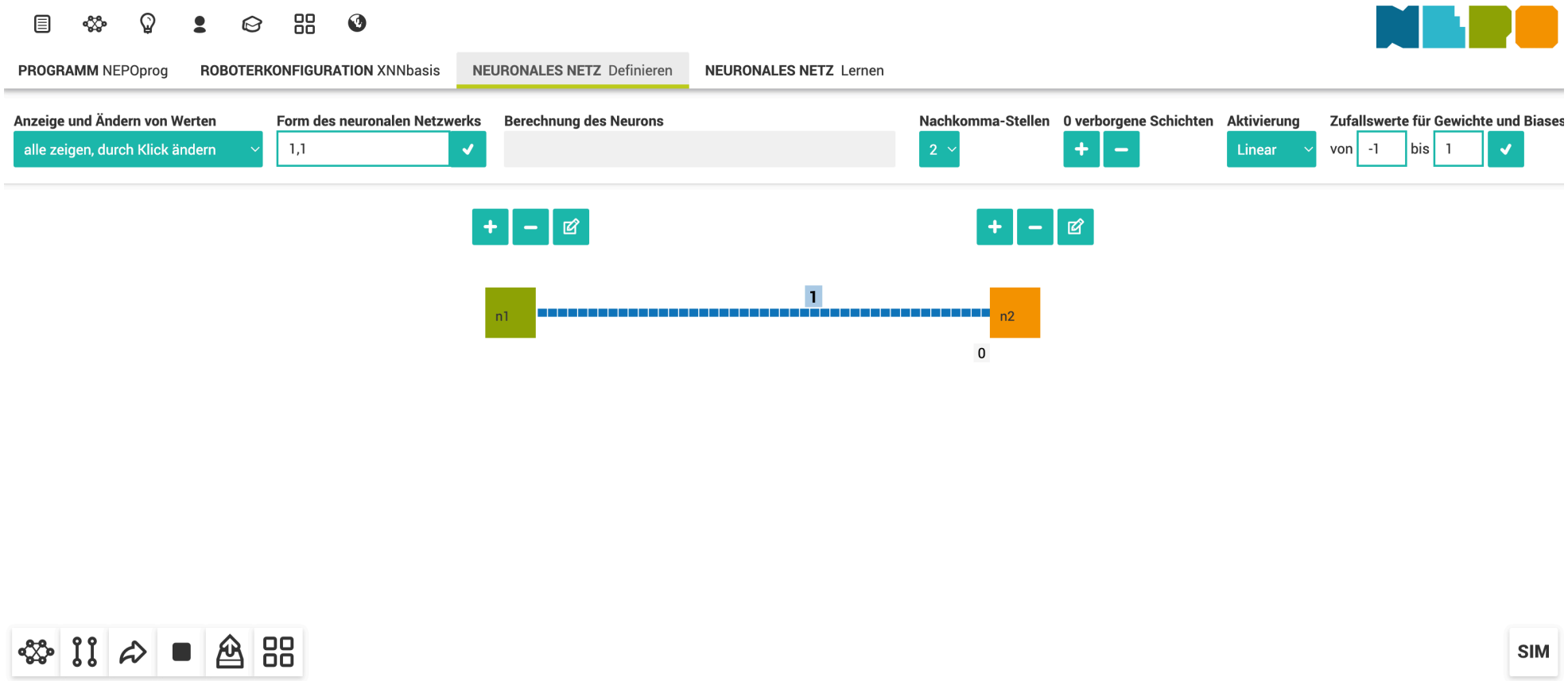


Fraunhofer IAIS

Open Roberta Lab

Neuronale Netze programmieren mit Open Roberta xNN

So sieht das voreingestellte Neuronale Netz aus.

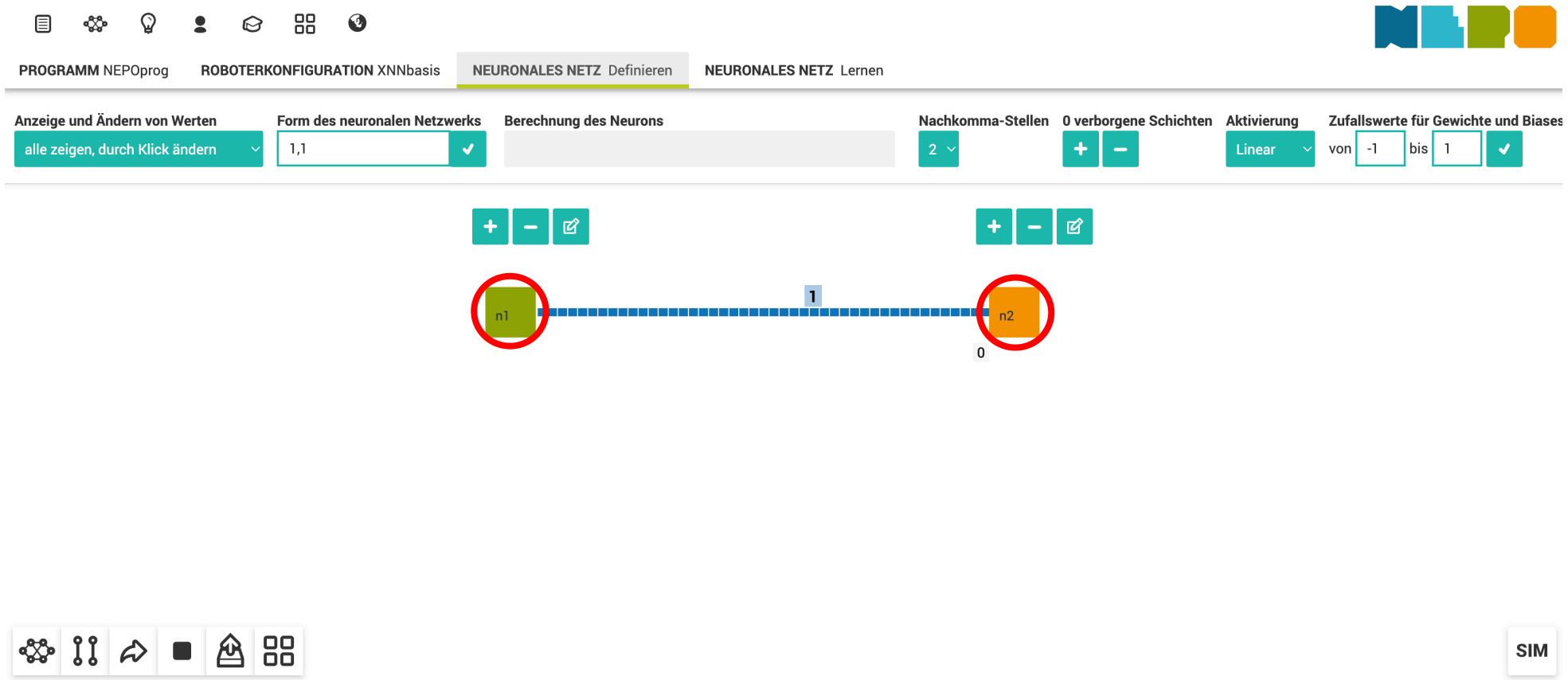


Fraunhofer IAIS

Open Roberta Lab

Neuronale Netze programmieren mit Open Roberta xNN

Mit einem Klick auf **+** können weitere Eingabe- bzw. Ausgabe-Neuronen erzeugt werden.

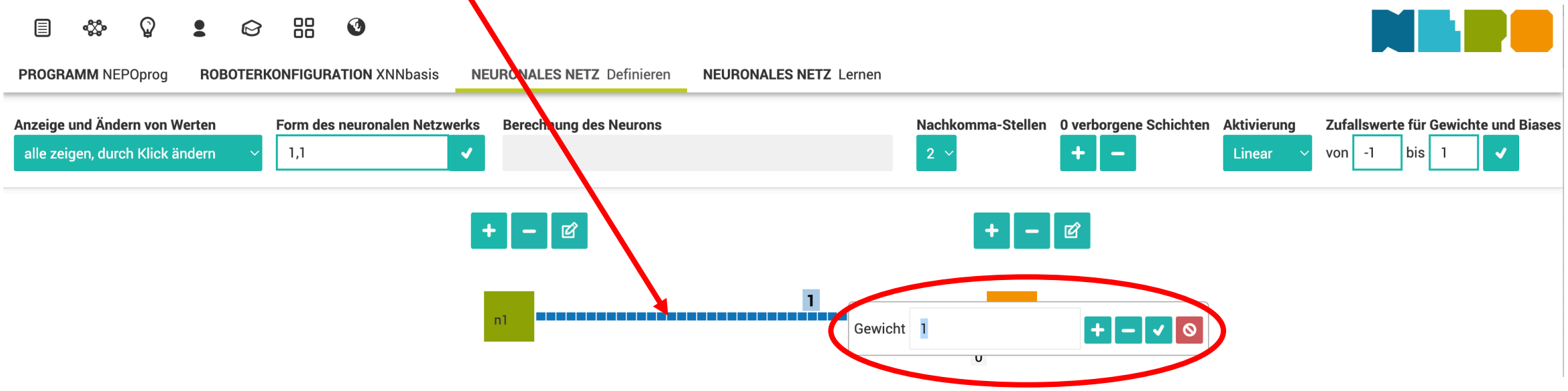


Fraunhofer IAIS

Open Roberta Lab

Neuronale Netze programmieren mit Open Roberta xNN

Mit einem Klick auf die gestrichelte Kante (hier: Verbindung zwischen Eingabe-Neuron *n1* und Ausgabe-Neuron *n2*) wird das Kantengewicht festgelegt.

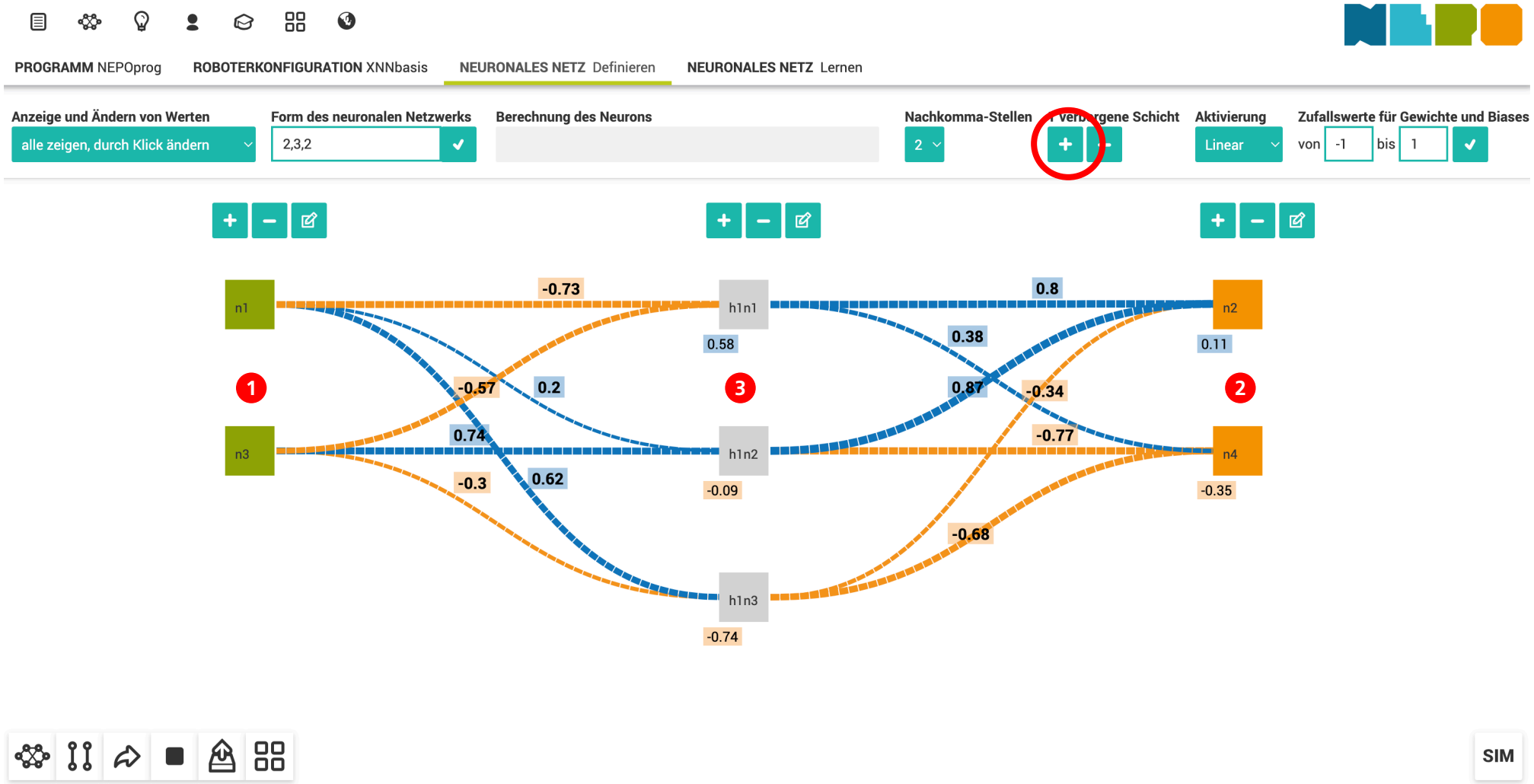


Fraunhofer IAIS

Neuronales Netz

Neuronale Netze programmieren mit Open Roberta xNN

1. Eingabe-Schicht
2. Ausgabe-Schicht
3. Verborgene Schicht (engl. hidden layer)

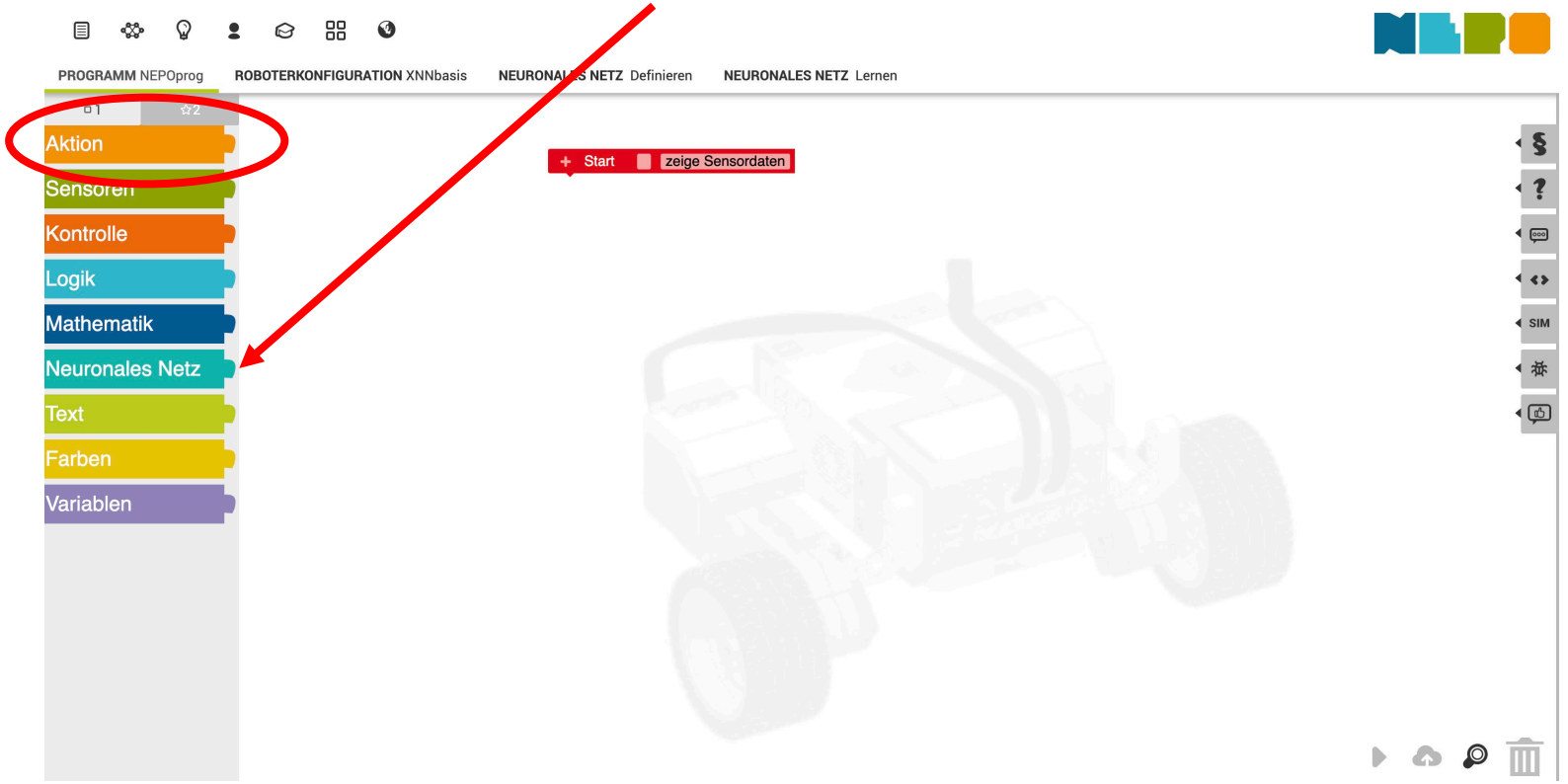


Fraunhofer IAIS

Open Roberta Lab

Neuronale Netze programmieren mit Open Roberta xNN

Durch einen Klick auf den Tab **Programm NEPOprog** gelangt man zurück in die Programmansicht. Die grundlegenden NEPO-Blöcke, die für das Neuronale Netz benötigt werden, befinden sich in der Kategorie **Neuronales Netz**. Mit diesen Blöcken kann das Neuronale Netz im Programm verwendet werden.



Fraunhofer IAIS

Open Roberta Lab

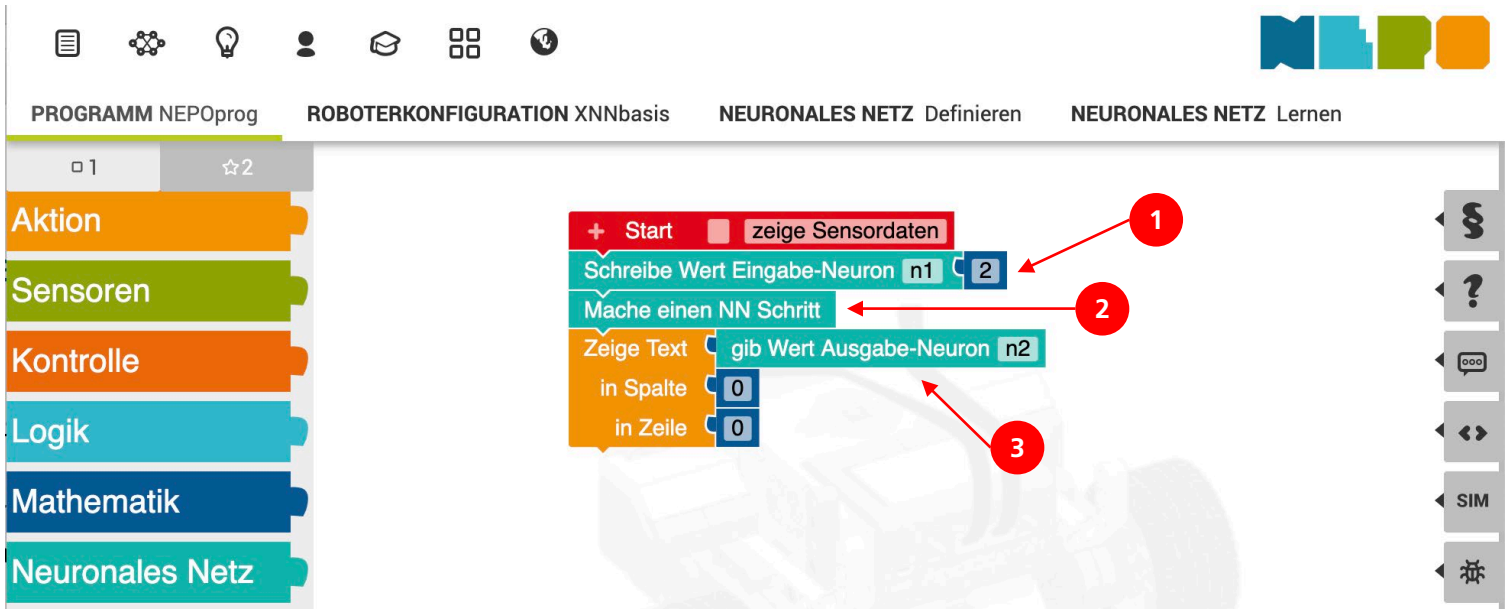
Neuronale Netze programmieren mit Open Roberta xNN

Alle Neuronen benötigen einen <u>eindeutigen</u> Namen.

1. Eingabe-Neuronen können konstante Werte, Variablen, Sensor-Werte oder auch Rückgabewerte aus Funktionen erhalten.

2. **Mit dem Block Mache einen NN Schritt wird die Berechnung im gesamten Neuronalen Netz durchgeführt**

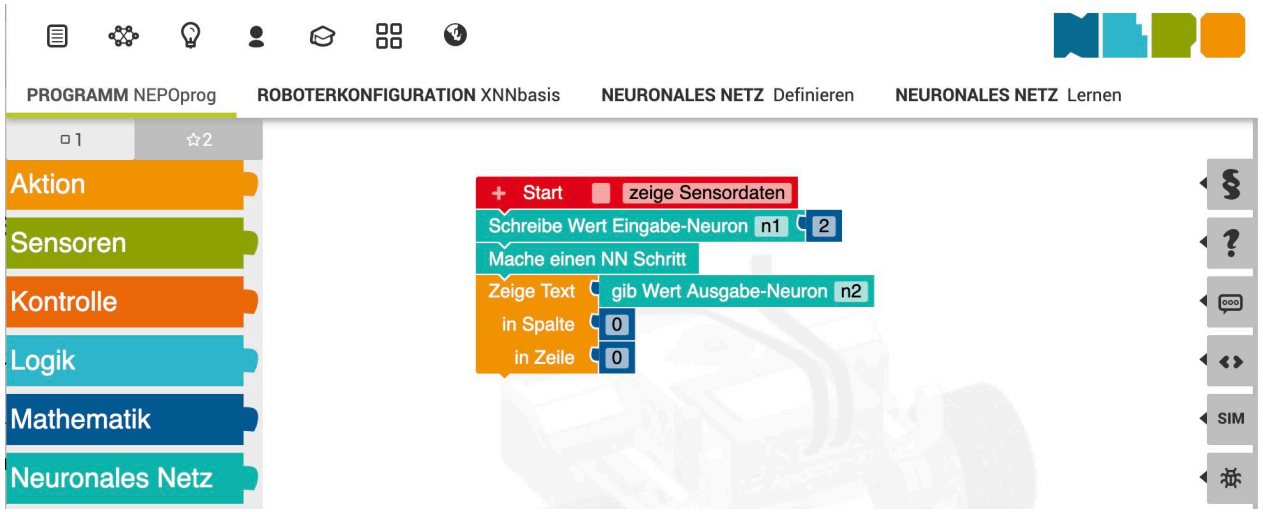
3. **Die im KNN ermittelten Werte können über den Block gib Wert Ausgabe-Neuron genutzt werden.**



Fraunhofer IAIS

Open Roberta Lab

Neuronale Netze programmieren mit Open Roberta xNN



Ablauf: Mit dem Block Schreibe Wert Eingabe-Neuron n1 wird ein konkreter Wert dem NN Eingabe Neuron n1 übergeben/zugewiesen. Dieser Wert steht nun im Neuronalen Netz zur Verfügung und wird durch den Block Mache einen NN Schritt im Neuronalen Netz berechnet. Mit diesem Block (Mache einen NN Schritt) wird das gesamte Neuronale Netz, unabhängig von der Anzahl der Neuronen, einmal von links nach rechts durchlaufen. Anschließend kann über den Block gib Wert Ausgabe-Neuron n2 der Wert aus dem Neuronalen Netz abgerufen / geholt werden. Gestartet wird das Programm in der Simulation (SIM). Klick auf den Start-Knopf.

Kapitel 02

Beispiele mit Open Roberta xNN

Fraunhofer IAIS

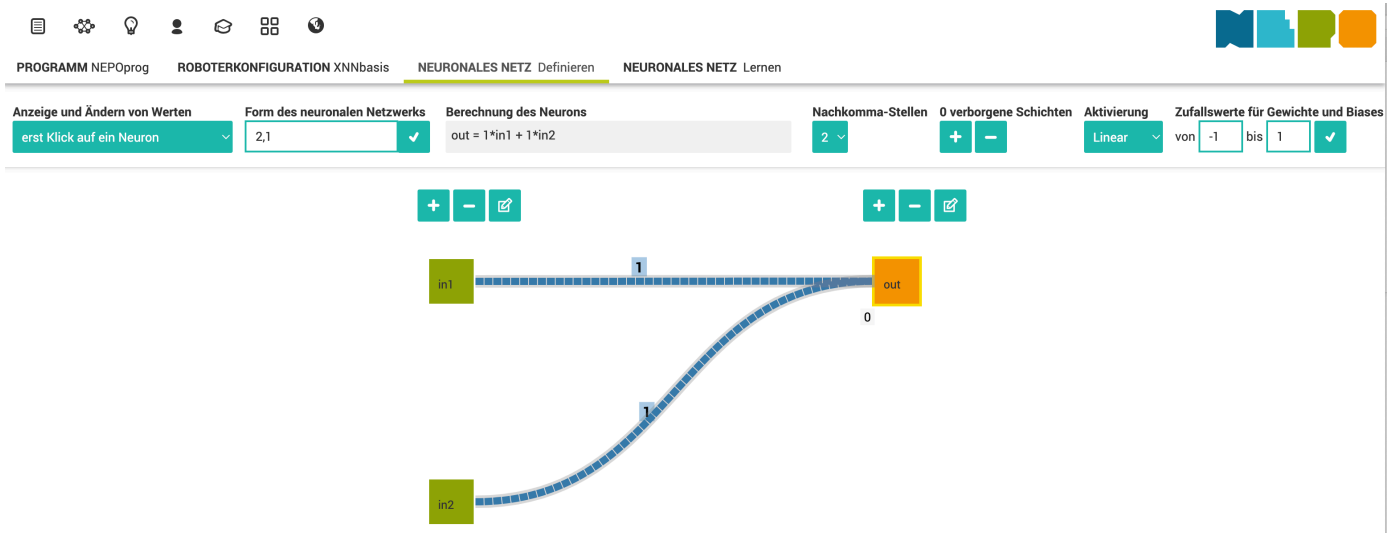
Neuronales Netz

Beispiele

Die folgenden Folien zeigen verschiedene Beispiele. Diese sollen veranschaulichen, wie die Werte im Neuronalen Netz berechnet werden.

Die Neuronalen Netzen im Open Roberta Lab sind **Feedforward-Netze** (das Netz wird mit jedem NN-Schritt einmal von links nach rechts durchlaufen). Die Berechnung der einzelnen Neuronen ist einfach.

Ein Klick auf das jeweilige Neuron (mit Ausnahme der Eingabe-Neuronen) zeigt, wie der Wert des Neurons berechnet wird.

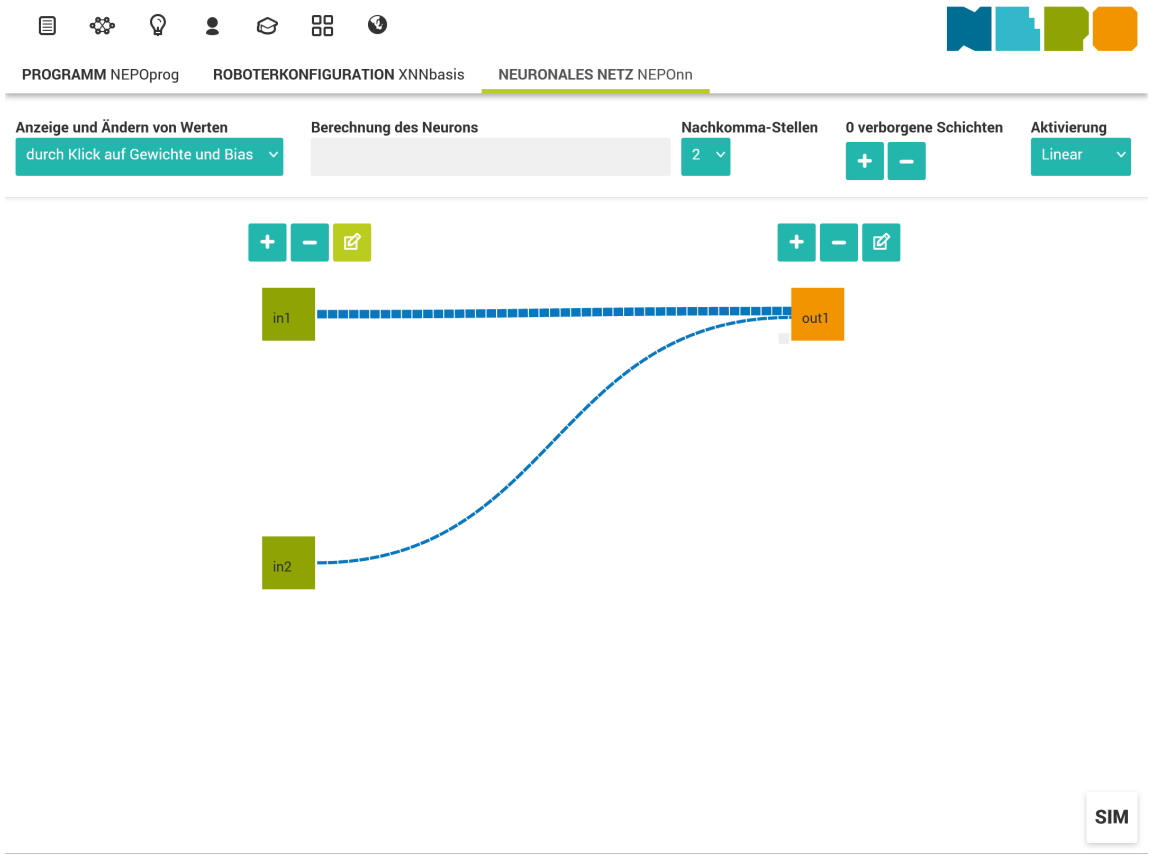


Fraunhofer IAIS

Neuronales Netz
Kantengewichtung

Die Stärke der Kante visualisiert die Gewichtung.

Die Kante zwischen Eingabe-Neuron *in1* und Ausgabe-Neuron *out1* ist stärker gezeichnet als die Kante zwischen Eingabe-Neuron *in2* und Ausgabe-Neuron *out1*, weil das Kantengewicht höher ist.

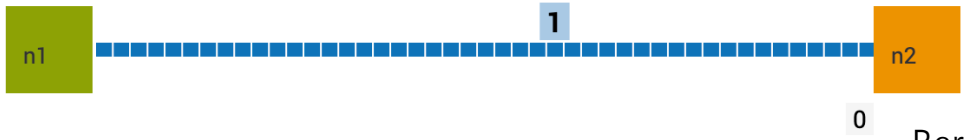


Fraunhofer IAIS

Neuronales Netz

Beispiele

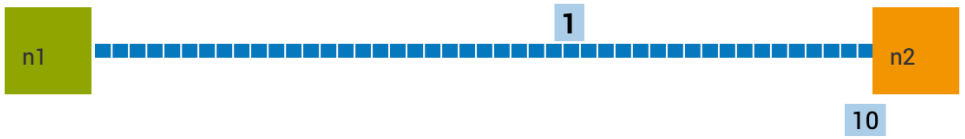
Eingabe: 1



Ausgabe: 1

Berechnung:
n2 = 1*n1

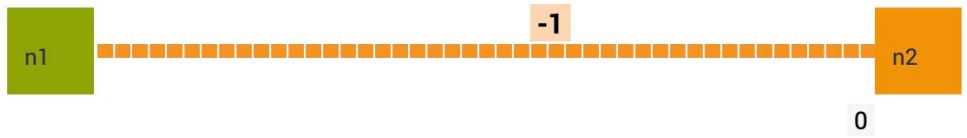
Eingabe: 1



Ausgabe: 11

Berechnung:
n2 = 10+1*n1

Eingabe: 1



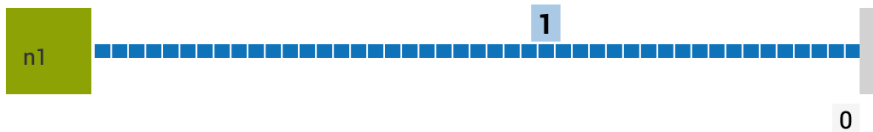
Ausgabe: -1

Berechnung:
n2 = 0 -1*n1

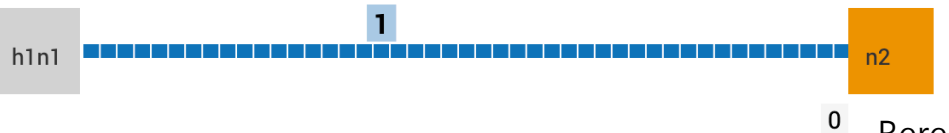
Neuronales Netz

Beispiele

Eingabe: 1



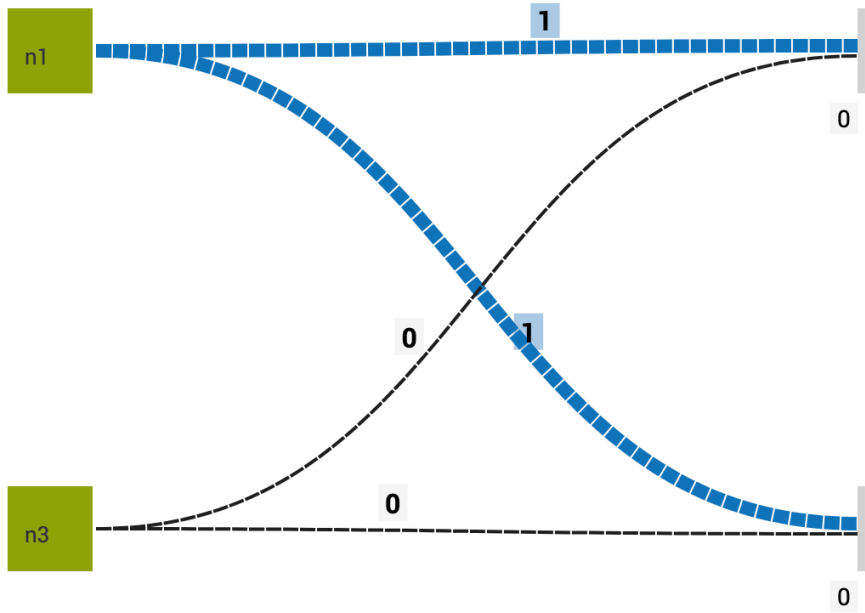
Berechnung:
h1n1= 1*n1

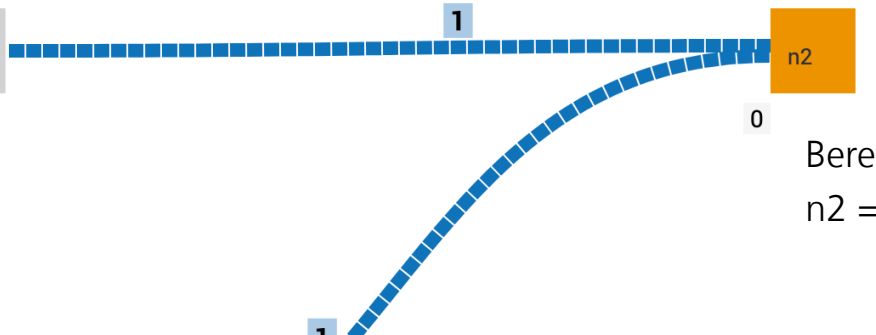


Ausgabe: 1

Berechnung:
n2 =1*h1n1

Eingabe: 1





Ausgabe: 2

Berechnung:
n2 = 1*h1n1 + 1*h1n2

Eingabe: 1

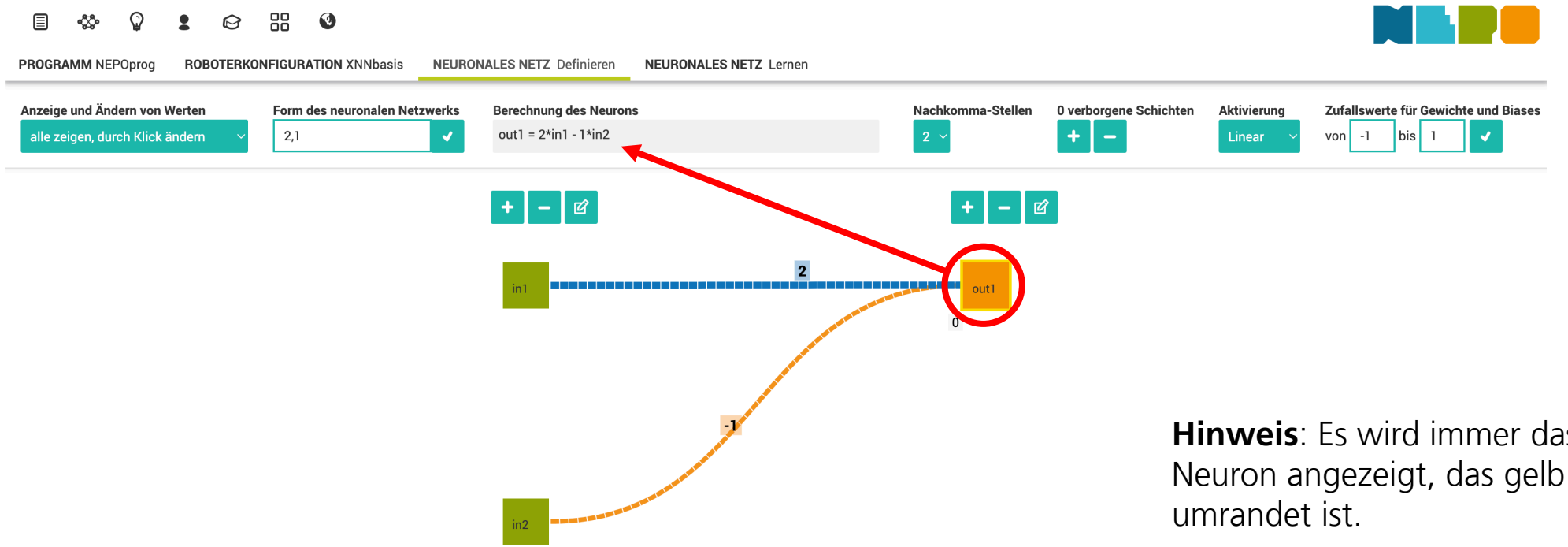
Fraunhofer IAIS

Neuronales Netz

Aufgabe – Tipp

Wie der Wert eines Neurons berechnet wird, sieht man, wenn das Ausgabe-Neuron hier: *out1* angeklickt wird.

Im Beispiel rechts: *out1* = 2 * *in1* – 1 * *in2*

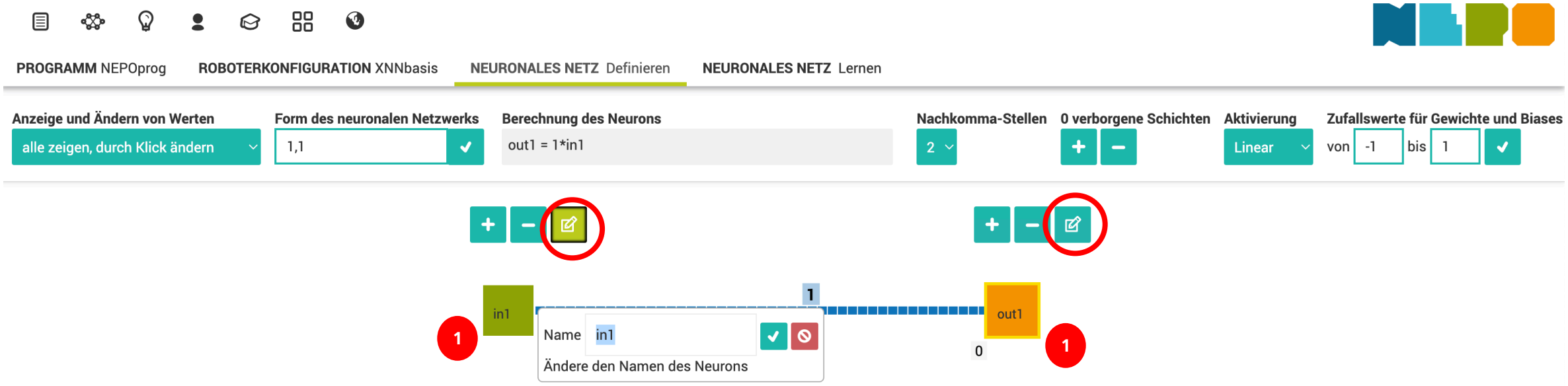


Hinweis: Es wird immer das Neuron angezeigt, das gelb umrandet ist.

Neuronales Netz

Änderung der Namen der Neuronen

1. Klick auf die Edit-Schaltfläche der Schicht, dann Klick auf das Neuron, dessen Namen geändert werden soll. Oder Doppelklick auf das Neuron.

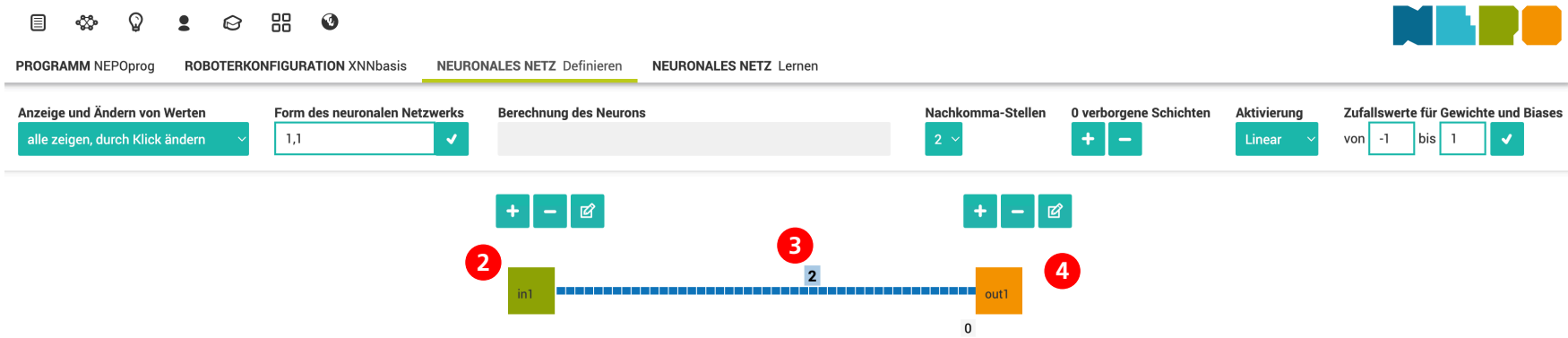


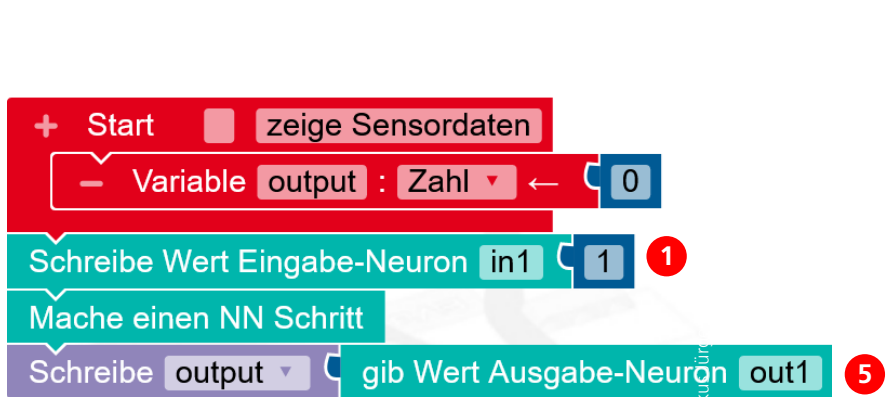
Fraunhofer IAIS

Neuronales Netz

Beispiel 1 – Erläuterung

1. Eingabe-Neuron *in1* erhält den Wert *1*
2. Diese stehen dem NN zur Verfügung
3. Die Gewichtung multipliziert den Wert mit *2*
4. Das Ausgabe-Neuron *out1* hat den Wert 2
5. Die Variable *output* erhält den Wert *2*

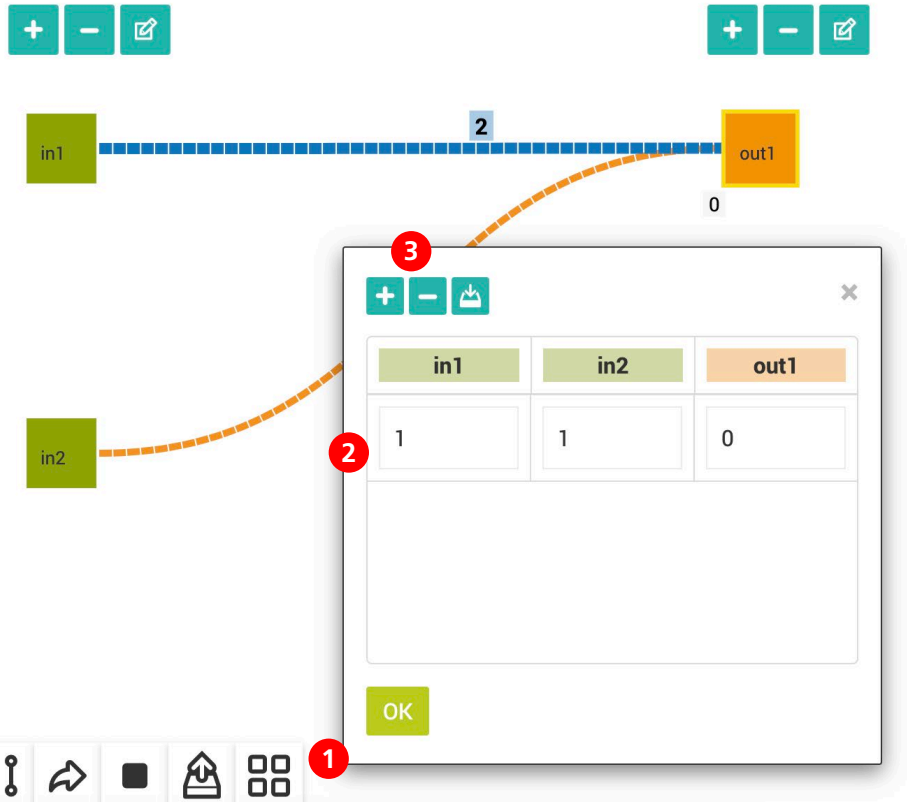




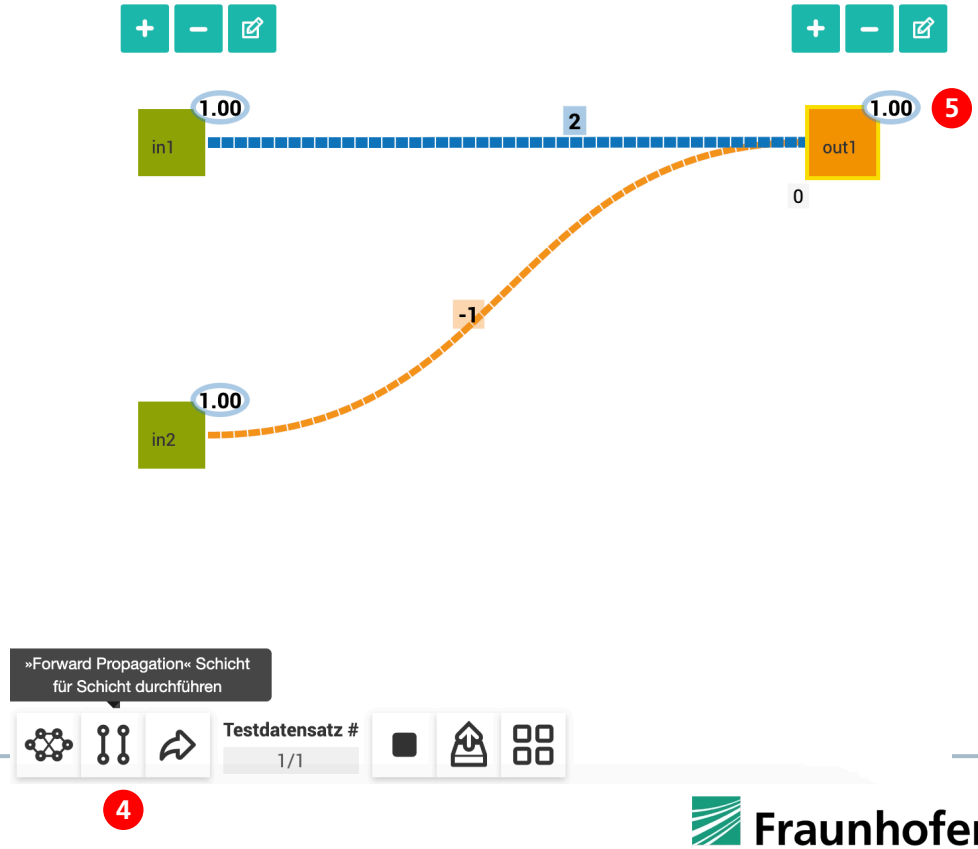
Neuronales Netz

Aufgabe – Lösung in der Ansicht „Definieren“

1. Klicke auf Test- /Trainingsdaten eingeben
2. Gib die Werte für *in1* = 1 und *in2* = 1 ein.
3. Entferne mit „–“ die nicht benötigten Zeilen und klicke auf „OK“.



4. Klicke auf „Forward-Propagation“
5. Das Ergebnis wird die am Ausgangs-Neuron out1 angezeigt.



Fraunhofer IAIS

Kapitel 03

Lernen mit xNN

Fraunhofer IAIS

Neuronale Netze

Lernen mit xNN

Mit Open Roberta xNN kann man das Neuronale Netz auch (via Backpropagation) lernen. Auf unserer Webseite www.open-roberta.org/xnn gehen wir ausführlich auf die Möglichkeiten des Lernens mit xNN ein. Es wird empfohlen, sich diese Seite vorab anzusehen.

Das nachfolgende Beispiel zeigt, wie das Neuronalen Netz trainiert wird, um die **Funktion f=2*x+1** zu lernen.

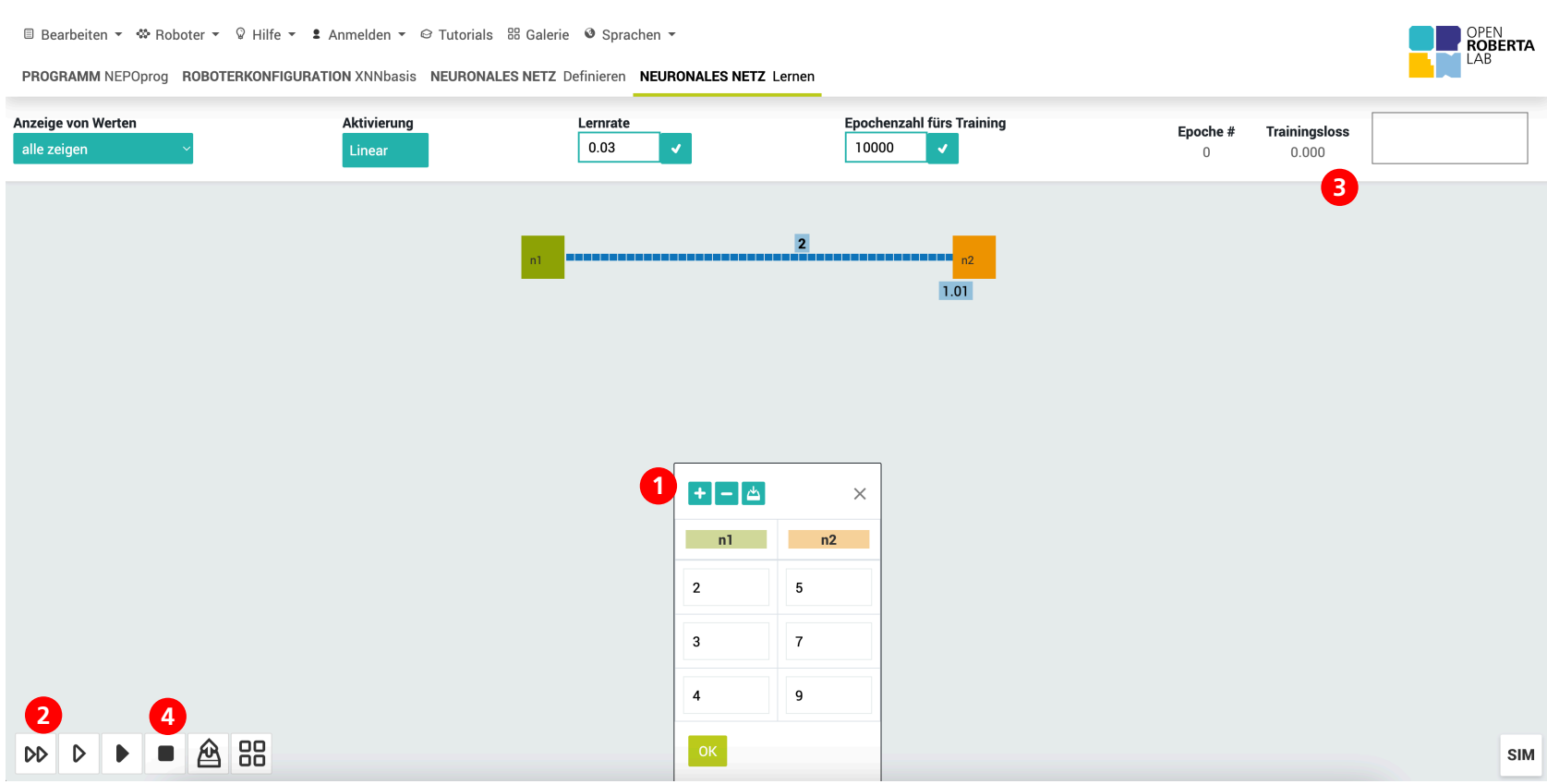
Als erstes Definieren wir ein Netz aus einem Eingangs-Neuron ***x*** und einem Ausgangs-Neuron ***y***. Mit einem Klick auf Zufallswerte werden zufallswerte beim Gewicht und beim Bias erzeugt.

- ✓ Beispiel: x=2 2*2+1 = 5
- ✓ Beispiel: x=3 2*3+1 = 7
- ✓ Beispiel: x=4 2*4+1 = 9

.

Neuronales Netz

Beispiel – Lernen mit xNN

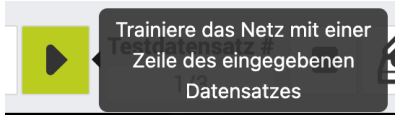


1. Eingeben der Trainingsdaten im Tab NEURONALES NETZ Lernen

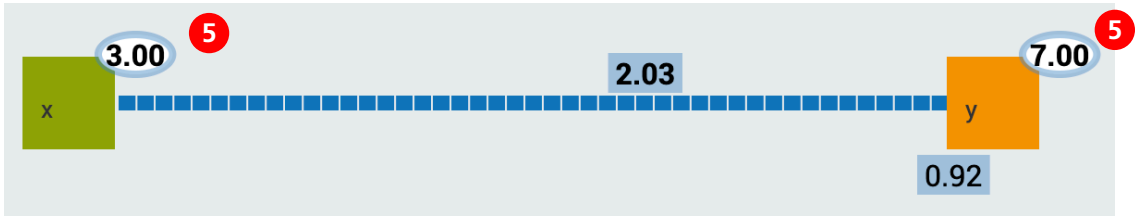
2. Nun beginnen wir das Netz zu trainieren

3. Ist der gewünschte Trainingsloss erreicht

   Kann das Training mit einem Klick auf 2 beendet werden.

4. Mit einem Klick auf 

Können die einzelnen Trainingsdaten und die vom Netz berechneten 5
Werte Schritt-für-Schritt aufgerufen werden.



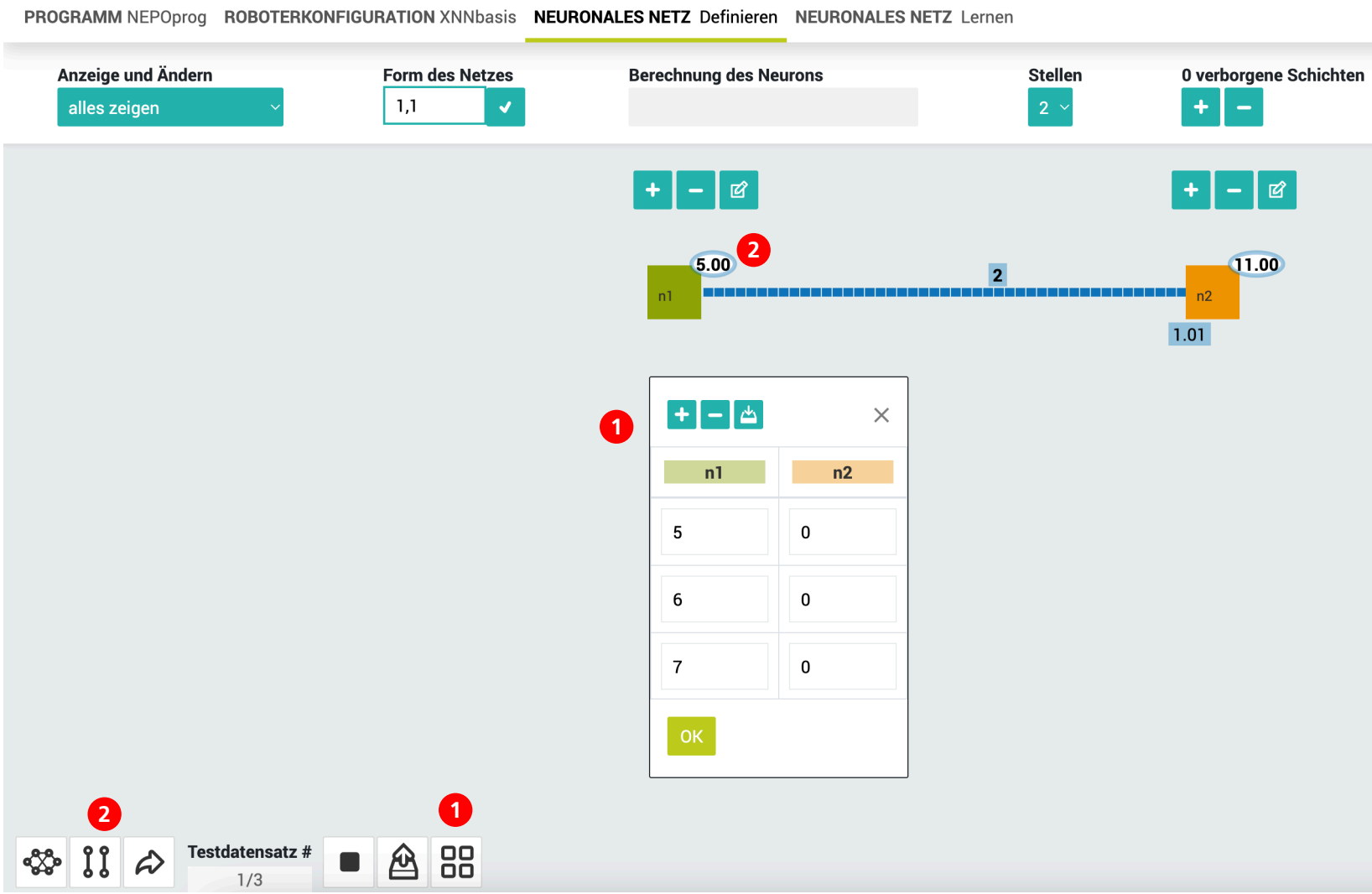
Neuronales Netz

Beispiel – Lernen mit xNN

Testen des trainierten Neuronalen Netzes

Nachdem das Netz trainiert wurde, können wir das Netz durch die Eingabe von Testdaten mit neuen Werten testen und somit prüfen, ob das Netz auch für Zahlenwerte funktioniert, mit denen es nicht trainiert wurde.

1. Eingabe von Testdaten
2. Schrittweise prüfen der Daten



Fraunhofer IAIS

Neuronale Netze

Training mit xNN – Glosar

Neuron:
Ein Neuron in einem neuronalen Netz ist die grundlegende Recheneinheit, die Informationen verarbeitet. Es wird auch als "künstliches Neuron" oder "Perzeptron" bezeichnet. Ein neuronales Netz besteht aus vielen solcher Neuronen, die in Schichten angeordnet sind. Das künstliche Neuron nimmt Eingaben entgegen (Eingabe-Neuron), multipliziert sie mit bestimmten Gewichten, addiert sie und wendet auf das Ergebnis eine Aktivierungsfunktion an. Die Aktivierungsfunktion bestimmt, ob und wie stark das Neuron aktiviert wird und welche Informationen es an die nächsten Neuronen weitergibt, bis die Information über Ausgabe-Neuronen zur Verfügung steht.

Aktivierungsfunktion:
Die Aktivierungsfunktion ist eine mathematische Funktion in einem neuronalen Netz, die die Ausgabe eines Neurons beeinflusst. Jedes Neuron in einem neuronalen Netz nimmt Eingaben entgegen, gewichtet sie und wendet dann eine Aktivierungsfunktion auf die gewichtete Summe an, um die Ausgabe zu erzeugen. Es gibt verschiedene Arten von Aktivierungsfunktionen: Lineare Funktion, Sigmoidfunktion, Tanh, ReLU (Rectified Linear Unit). Für die Arbeit mit Open Roberta Lab bietet sich die lineare Funktion an. Sie führt bei einfachen Aufgaben zu guten Ergebnissen und eignet sich aufgrund der leichten Erklärbarkeit besonders für den Einstieg.

Bias:
Bias ein Wert, der zu den Eingaben und Gewichtungen eines Neurons hinzugefügt wird, bevor er durch eine Aktivierungsfunktion geht. Dieser zusätzliche Wert ermöglicht es dem Neuron, auch dann aktiviert zu werden, wenn alle Eingabewerte null sind. Bei manchen Aktivierungsfunktionen (z.B. ReLU) kann der Bias auch eine „hemmende Wirkung“ haben. Hier wird das Neuron erst dann aktiviert, wenn ein bestimmter negativer „Schwellenwert“ überschritten wird.

Neuronale Netze

Training mit xNN – Glosar

(Kanten)Gewicht
Gewichte sind Zahlen, die bestimmen, wie wichtig oder unwichtig die Verbindungen zwischen den Neuronen sind. Durch die Anpassung dieser Gewichte lernt das Netz, Daten richtig zu verarbeiten und Muster zu erkennen.
Während des Trainings (des Lernens) des neuronalen Netzes werden diese Gewichte angepasst, um das Netz dazu zu bringen, die gewünschten Ausgaben für bestimmte Eingaben zu erzeugen. Der Lernprozess besteht darin, die Gewichte so zu optimieren, dass das Netz die richtigen Muster und Beziehungen in den Daten erkennt.

Verborgene Schicht
Ein Neuronales Netz kann in Schichten eingeteilt werden. Die erste Schicht erhält Eingaben (Werte) über die Eingangs-Neuronen, die letzte gibt Ausgaben (Werte die als Wahrscheinlichkeiten interpretiert werden, Ausgabe-neuronen) aus. Dazwischen gibt es "versteckte bzw. Verborgene Schichten", weil wir ihre Arbeit nicht direkt sehen können. Diese verborgenen Schichten lernen, komplexe Muster in den Daten zu verstehen, indem sie Eingaben miteinander kombinieren. Sie sind wichtig, um schwierige Probleme zu lösen. Die Anzahl und Größe dieser versteckten Schichten beeinflussen, wie gut das Netzwerk funktioniert.

Neuronale Netze

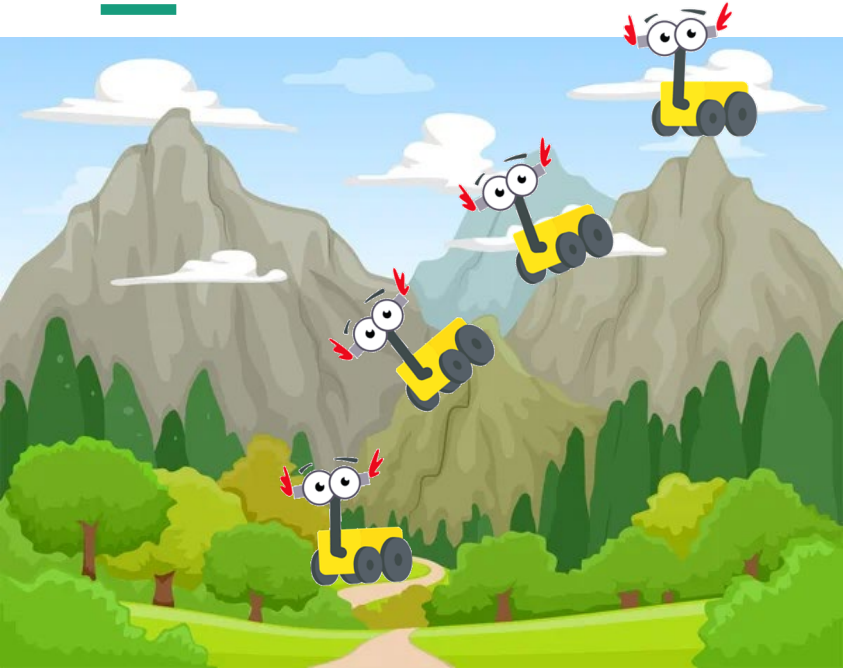
Training mit xNN – Glosar

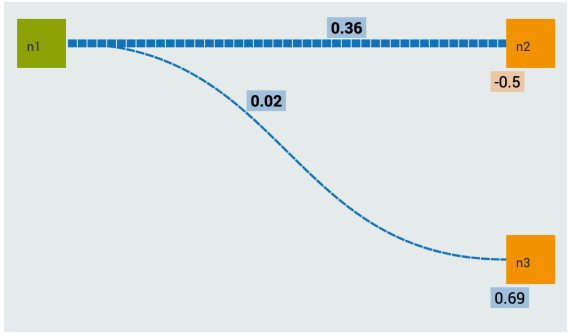
Trainingsloss:
Das Trainingsloss (auch Trainingsverlust genannt) ist ein Maß dafür, wie gut ein neuronales Netz während des Trainings Aufgaben löst. Es misst den Unterschied zwischen den vorhergesagten Ausgaben des Netzes und den tatsächlichen Zielwerten (auch Labels genannt) für die Trainingsdaten. Trainingsloss eine Art "Fehler", der angibt, wie weit die Vorhersagen des neuronalen Netzwerks von den tatsächlichen Werten entfernt sind. Während des Trainingsprozesses wird versucht, dieses Loss zu minimieren, indem die Gewichtungen und Bias-Werte des Netzes angepasst werden.

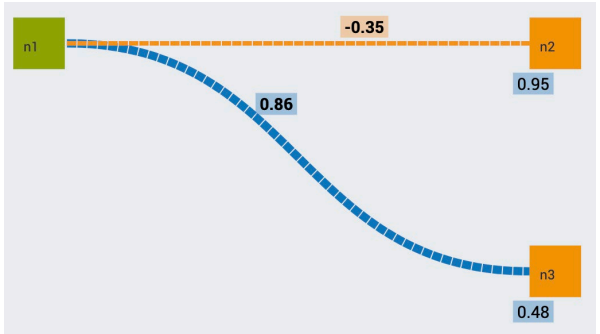
Epoche:
In Bezug auf neuronale Netze bezeichnet eine "Epoche" einen vollständigen Durchlauf durch alle Trainingsdaten während des Trainingsprozesses. In anderen Worten, während einer Epoche sieht das neuronale Netz jeden Trainingsdatensatz einmal.

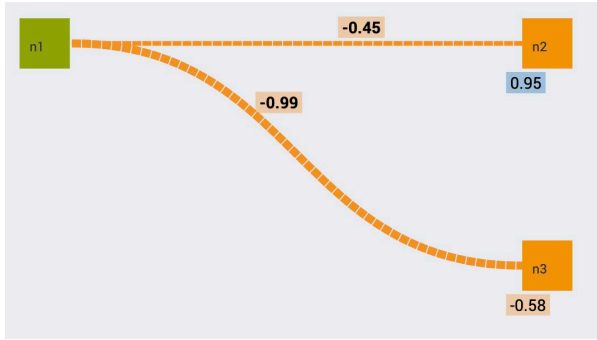
Backpropagation:
Backpropagation ist ein Lernverfahren, das im künstlichen neuronalen Netz verwendet wird. Es hilft dem Netz, aus Fehlern zu lernen und seine Leistung zu verbessern. Die Anpassung der Gewichte erfolgt mit einem Optimierungsalgorithmus, der sicherstellt, dass das Netz die Gewichtungen und Bias-Werte so ändert, dass der Fehler reduziert wird.

Die Ausgangslage entscheidet mit über das Training eines Neuronalen Netzes, sprich welcher Weg ins Tal genommen wird. Die Gewichte im Neuronalen Netz entsprechen den Geokoordinaten (Osten, Süden, Westen, Norden) und die Höhe der Kostenfunktion (bei Open Roberta Trainingsloss genannt).

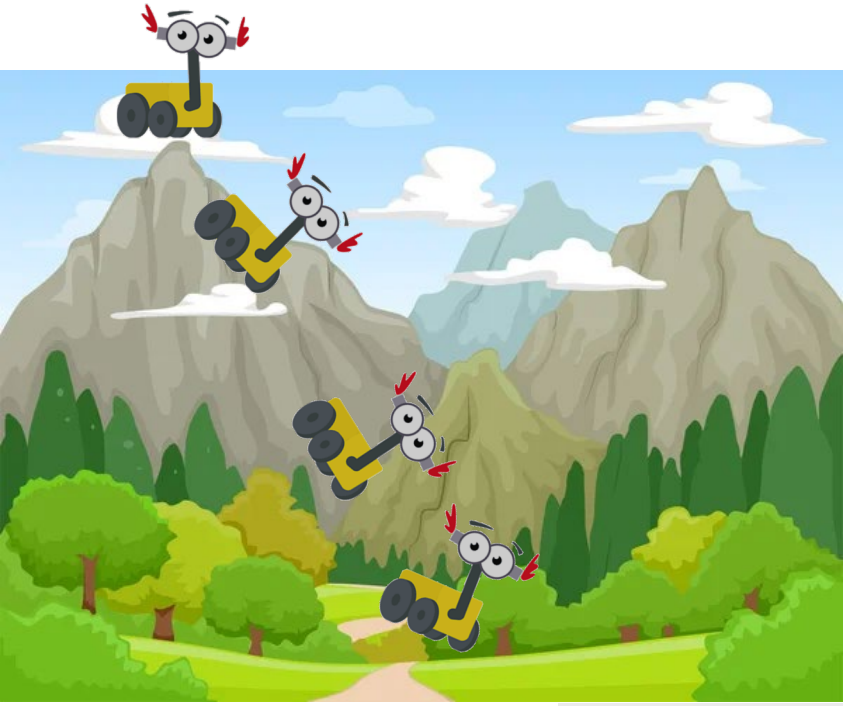




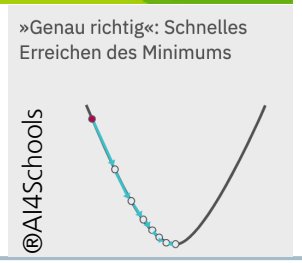




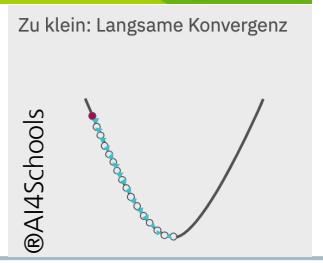
Die Lernrate bestimmt, wie schnell ein Minimum gefunden werden kann.
Ist diese sehr klein (Beispiel 2) dauert es lange, ist die Lernrate dagegen zu groß (Beispiel 3) kann ggf. kein Minimum gefunden werden. Der Abstieg vom „Gipfel“ hinab ins Tal wird **Gradientenabstieg (englisch: Gradient Descent) genannt und ist ein Optimierungsalgorithums der beim maschinellen Lernen häufig verwendet wird.**



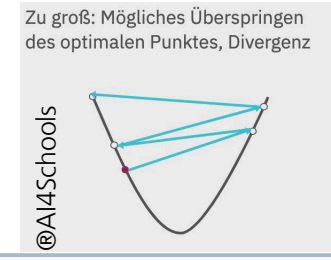
Lernrate = 0,003



Lernrate = 0,00003



Lernrate = 0,3



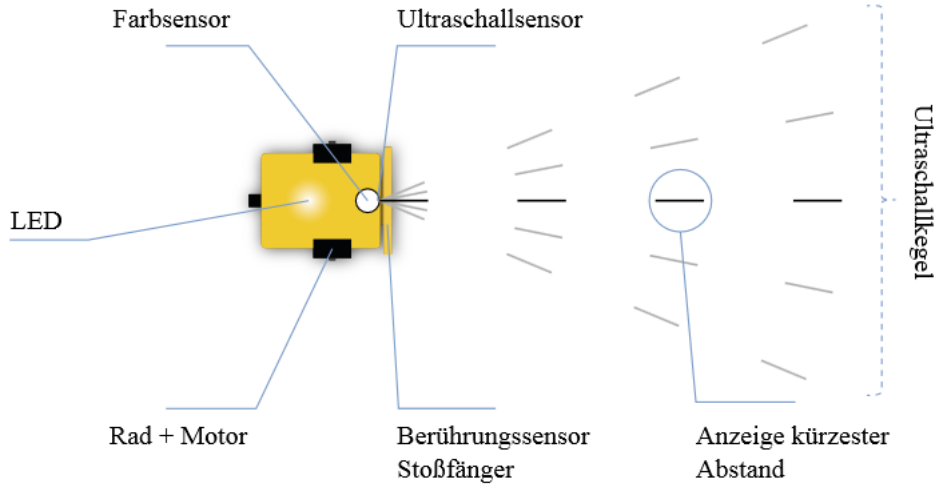
Kapitel 03

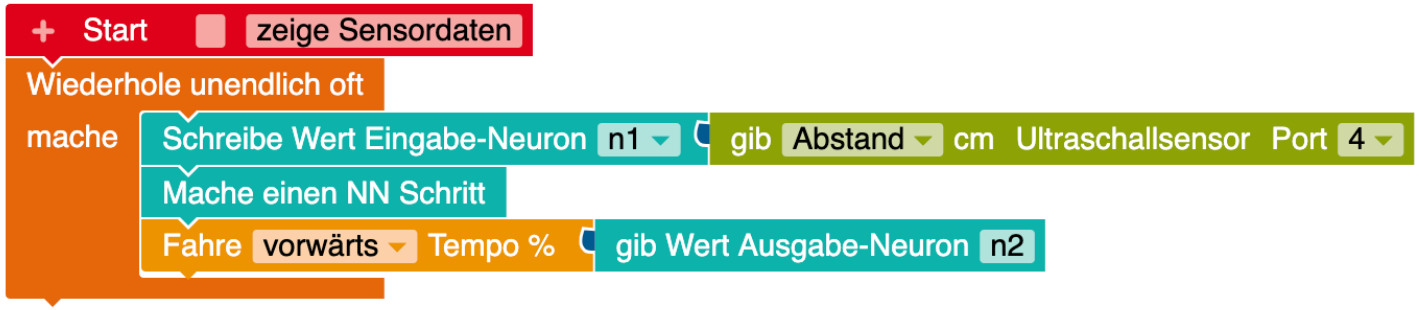
xNN und Roboter

Neuronales Netz

Ausblick – Roboter mit Open Roberta xNN programmieren

Einem Eingabe-Neuron werden in der Regel Sensorwerte übergeben. Auf diese Weise reagiert das neuronale Netz auf Informationen von "außen". Ein Beispiel dafür ist der Ultraschallsensor. Der Ultraschallsensor befindet sich in der Mitte des Roboters.



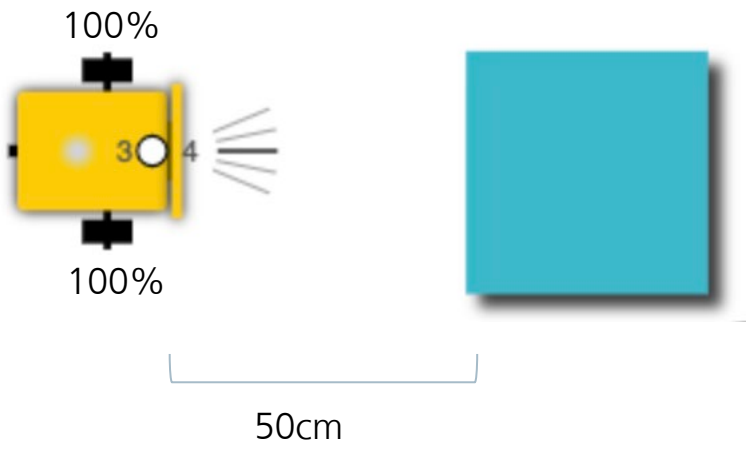


Fraunhofer IAIS

Neuronales Netz

Beispiel Lernen - Abstand einhalten

Dieses Beispiel zeigt, wie das Netz trainiert werden kann, damit ein Roboter mit einem Abstandssensor (Ultraschallsensor) einen „Mindestabstand“ zu einem Objekt einhält.



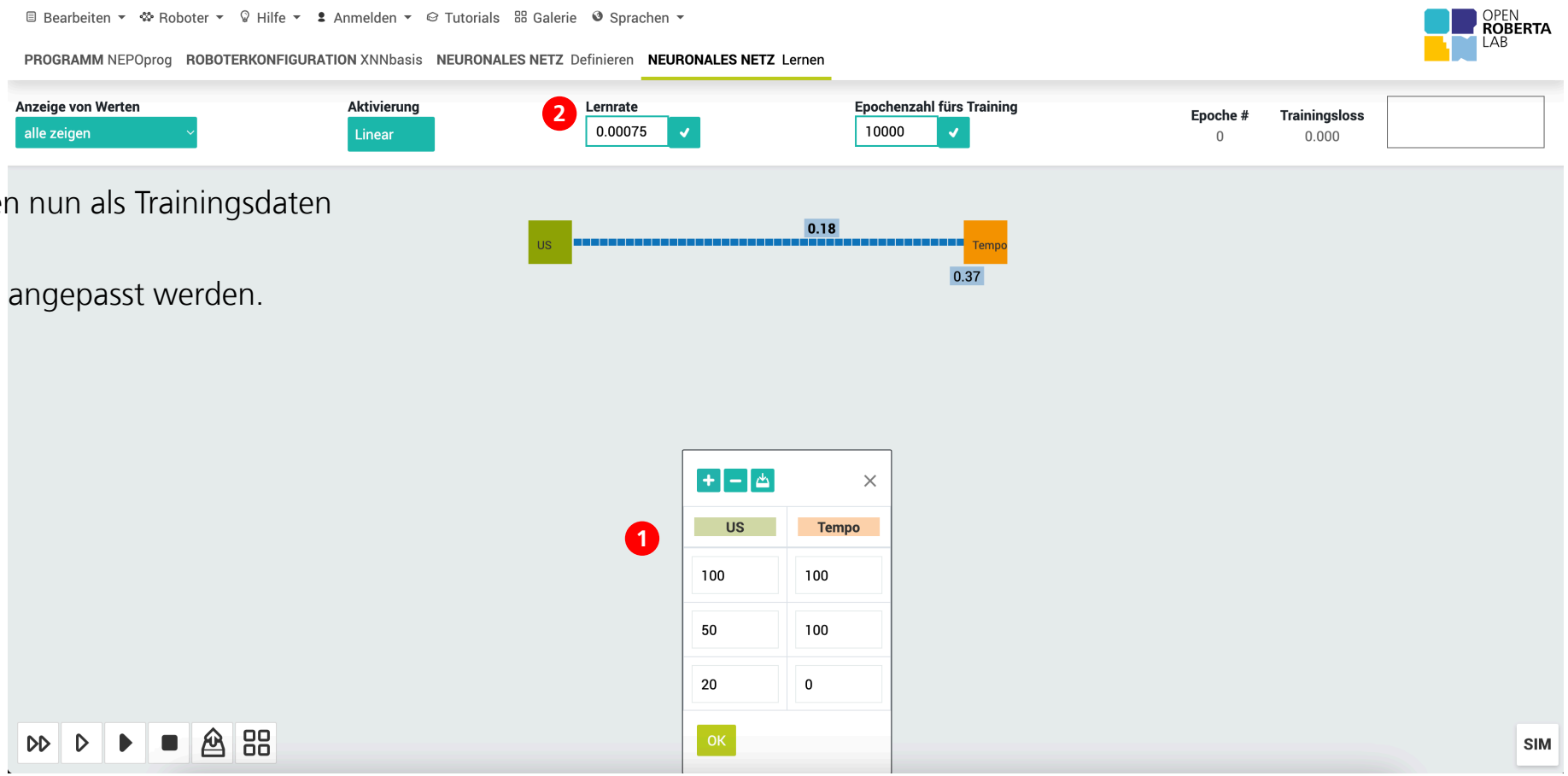
- Mit dem US-Sensor misst der Roboter den Abstand zu Objekten, die sich in Fahrtrichtung befinden.
- Die folgende Tabelle zeigt, bei welchen gemessenen Distanzwerten der Roboter welche Geschwindigkeit fahren soll. Negative Werte werden vom Roboter als Rückwärtsfahrt interpretiert.

Abstand in cm	Tempo in % der maximalen Geschwindigkeit des Roboters
100	100
50	100
20	-50
10	-100

Neuronales Netz

Beispiel Lernen - Abstand einhalten

1. Die Werte aus der Tabelle können nun als Trainingsdaten eingegeben werden.
2. Die Lernrate muss entsprechend angepasst werden.



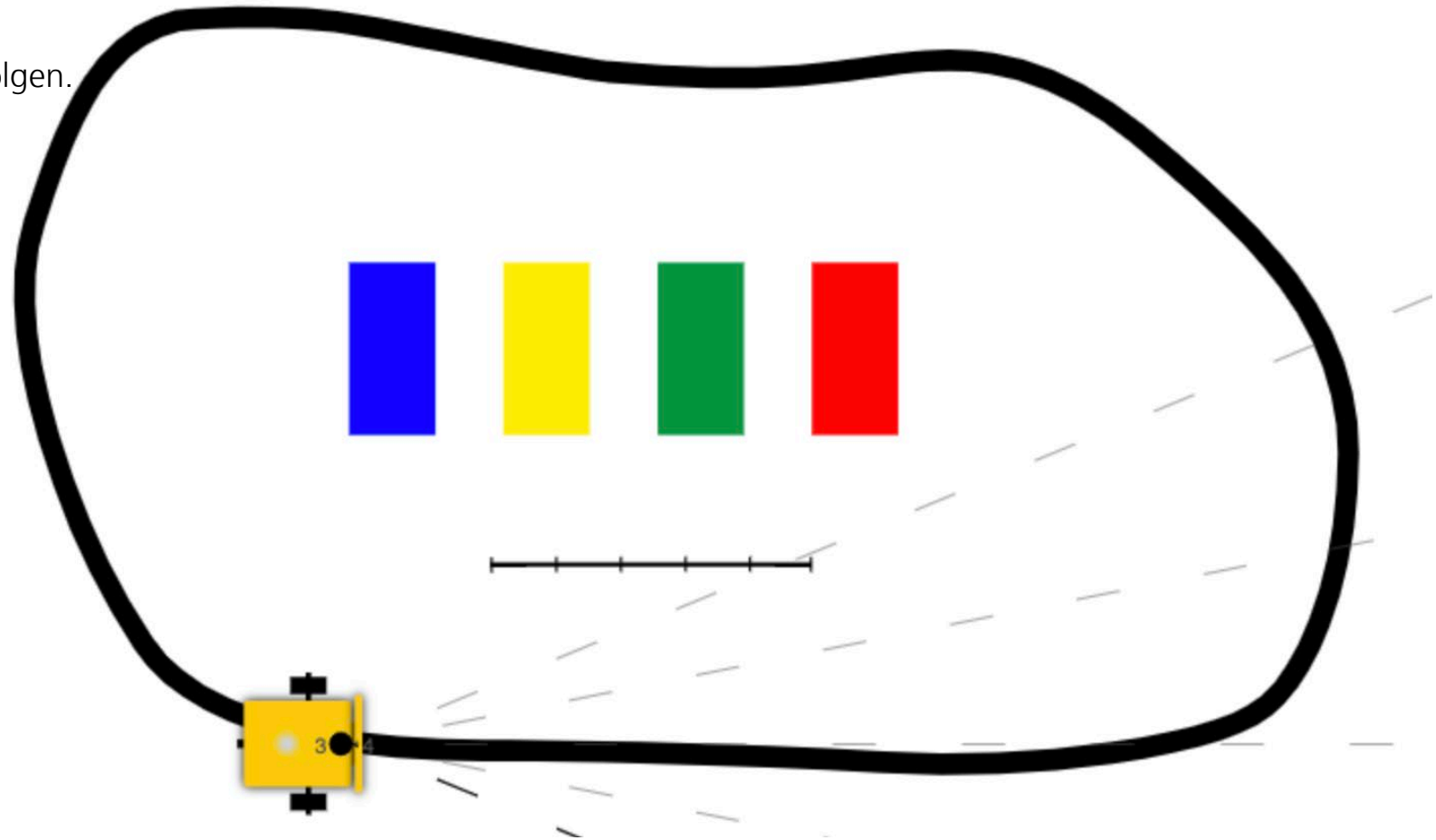
Fraunhofer IAIS

Neuronales Netz

Folgen einer schwarzen Linie

Beispiel: Unser Roboter soll einer schwarzen Linie folgen.

Frage: Welche Daten stehen uns zur Verfügung?

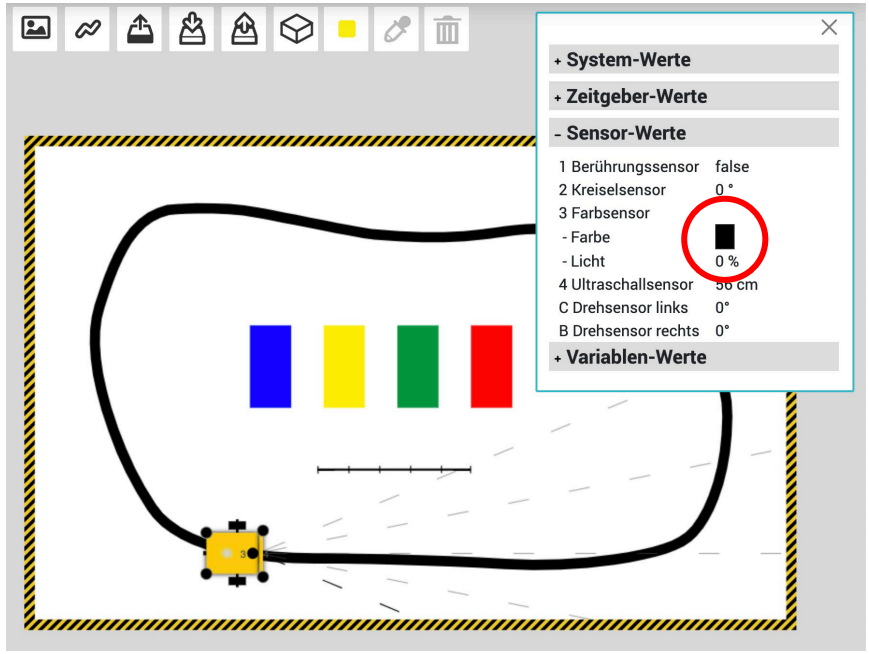


Neuronales Netz

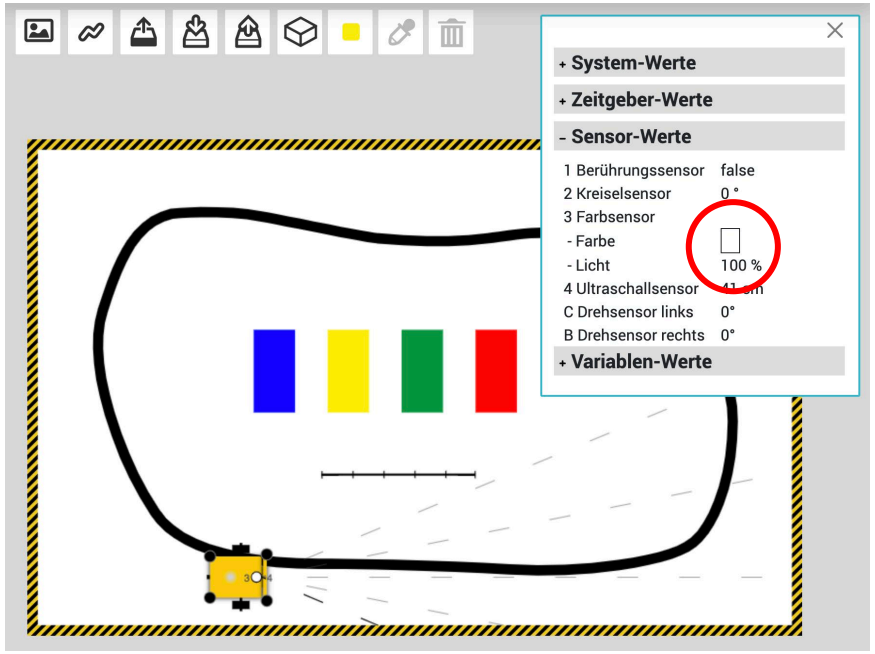
Folgen einer schwarzen Linie

Da unser Roboter einen Farbsensor hat, können wir die gemessenen Lichtwerte nutzen.

Wenn der Roboter auf der schwarzen Linie steht, erhalten wir als Lichtwert 0%.



Steht der Roboter auf einer weißen Fläche, erhalten wir als Lichtwert 100%.

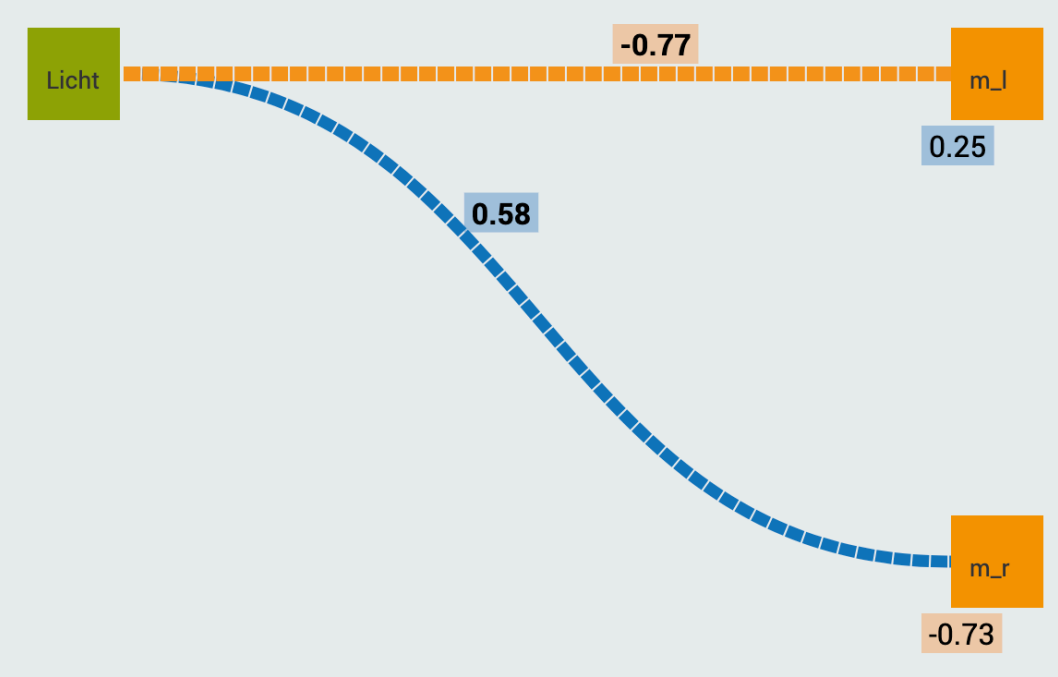


Probieren wir aus, ob diese beiden Werte bereits ausreichen, um unseren Roboter einer schwarzen Linie folgen zu lassen.

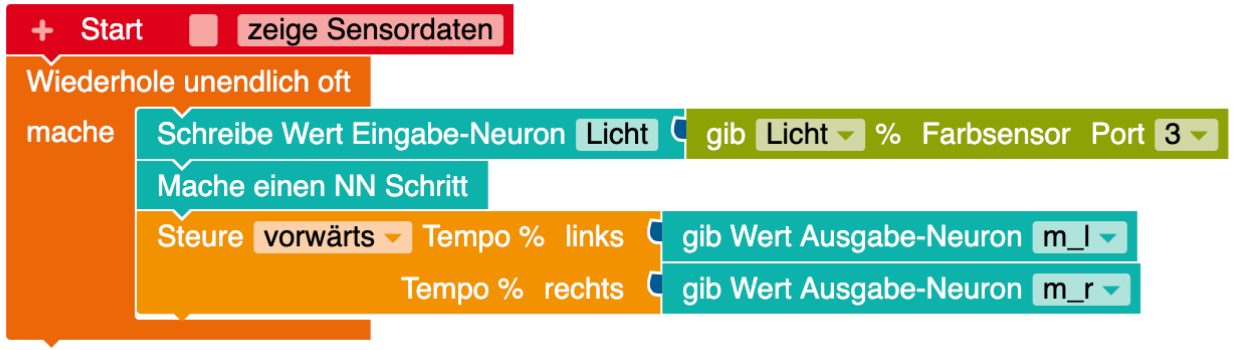
Neuronales Netz

Folgen einer schwarzen Linie

Wir erzeugen ein Netz mit einem Eingangs-Neuron für den Lichtsensor und zwei Ausgangs-Neuronen für die beiden Motoren. Weiterhin belegen wir das Netz mit Zufallswerten.



Entsprechend dem Netz programmieren wir ein Programm, das den Wert des Lichtsensors in das Netz einspeist und die Werte der Ausgangsneuronen an die beiden Motoren weiterleitet.



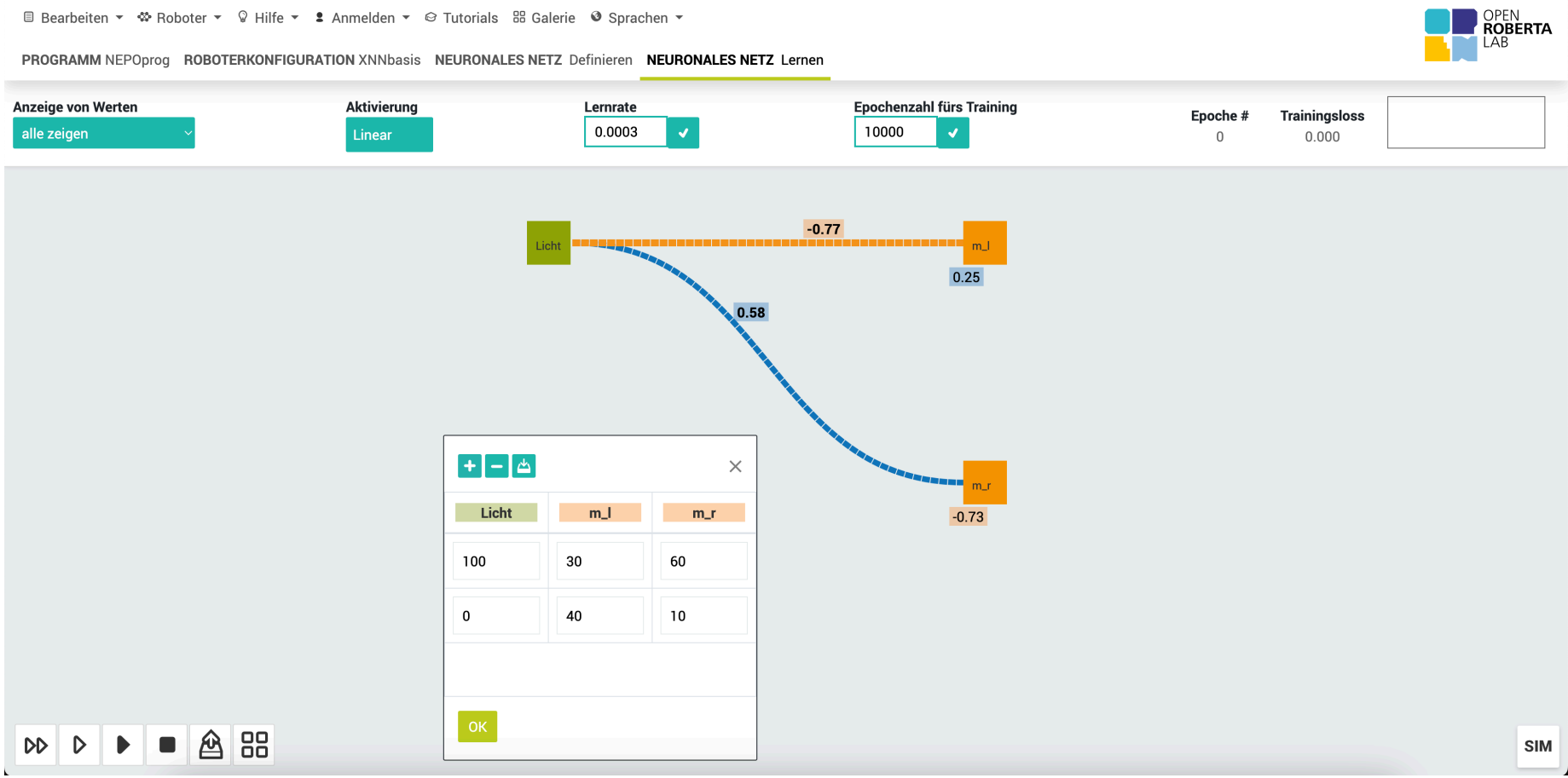
Fraunhofer IAIS

Neuronales Netz

Folgen einer schwarzen Linie

Nun geben wir die Werte als Trainingsdaten in das Netz ein, stellen die Lernrate ein und trainieren das Netz.

.

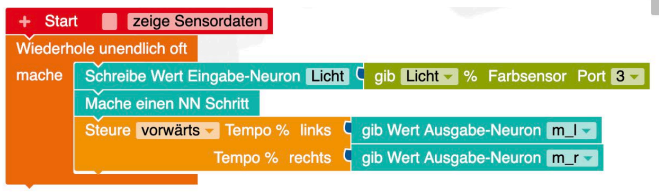


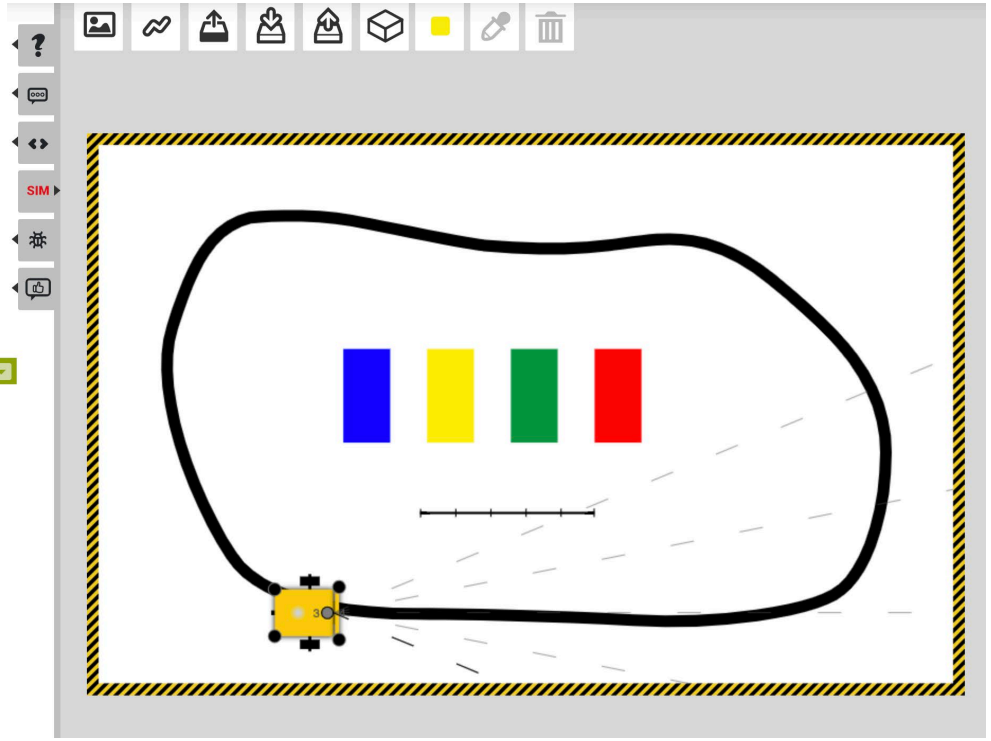
Fraunhofer IAIS

Neuronales Netz

Folgen einer schwarzen Linie

Bei einem Traingsloss von ca. 20 kann getestet werden, ob der Roboter bereits der Linie folgt. Ist dies nicht der Fall, kann das Training einfach fortgesetzt werden.





Fraunhofer IAIS

Neuronales Netz

Folgen einer schwarzen Linie

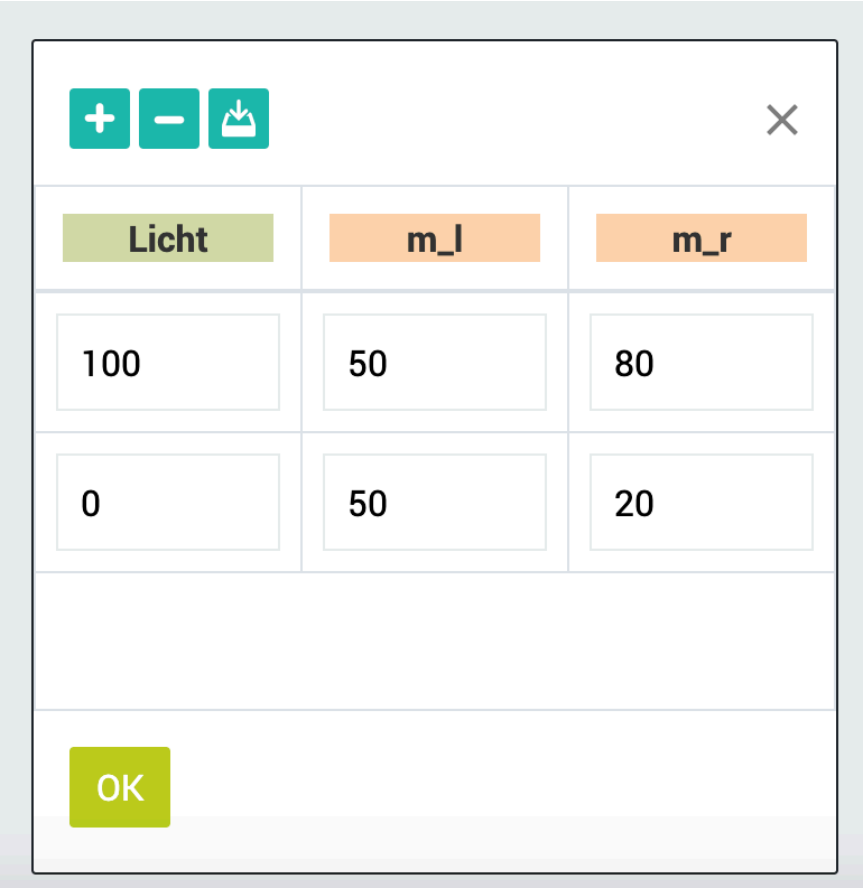
Weiterhin können die Trainigsdaten angepasst werden.

Beispielsweise können die Werte für die Motordrehzahl erhöht werden.

Anschließend muss das Netz natürlich neu trainiert werden. Hier kann man einfach das bestehende Netz weiter trainieren.

Interessant ist es hier, mit den Werten so zu experimentieren, dass der Roboter möglichst schnell auf der Linie fährt, aber die Linie nicht verliert.

Frage:
Gibt es einen Wert, den man verwenden kann, damit der Roboter sehr schnell entlang der Linie fahren kann?

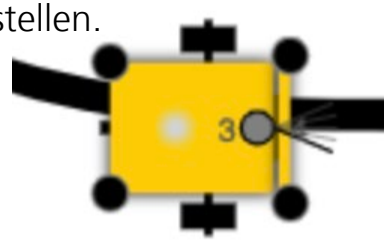


Fraunhofer IAIS

Neuronales Netz

Folgen einer schwarzen Linie

Antwort: Ja, dazu den Roboter auf die Kante zwischen schwarz und weiß stellen.



1

Dieser Wert kann wie folgt für die Trainingsdaten verwendet werden.

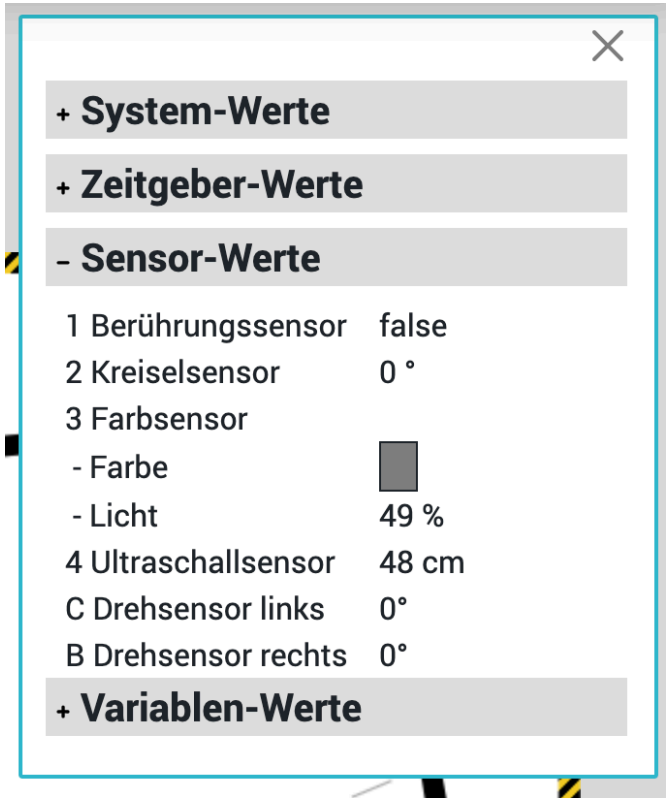
3

Licht	m_l	m_r
100	50	80
0	50	20
50	60	60

OK

Hier misst der Lichtsensor einen Grauwert zwischen 40 und 80%.

2

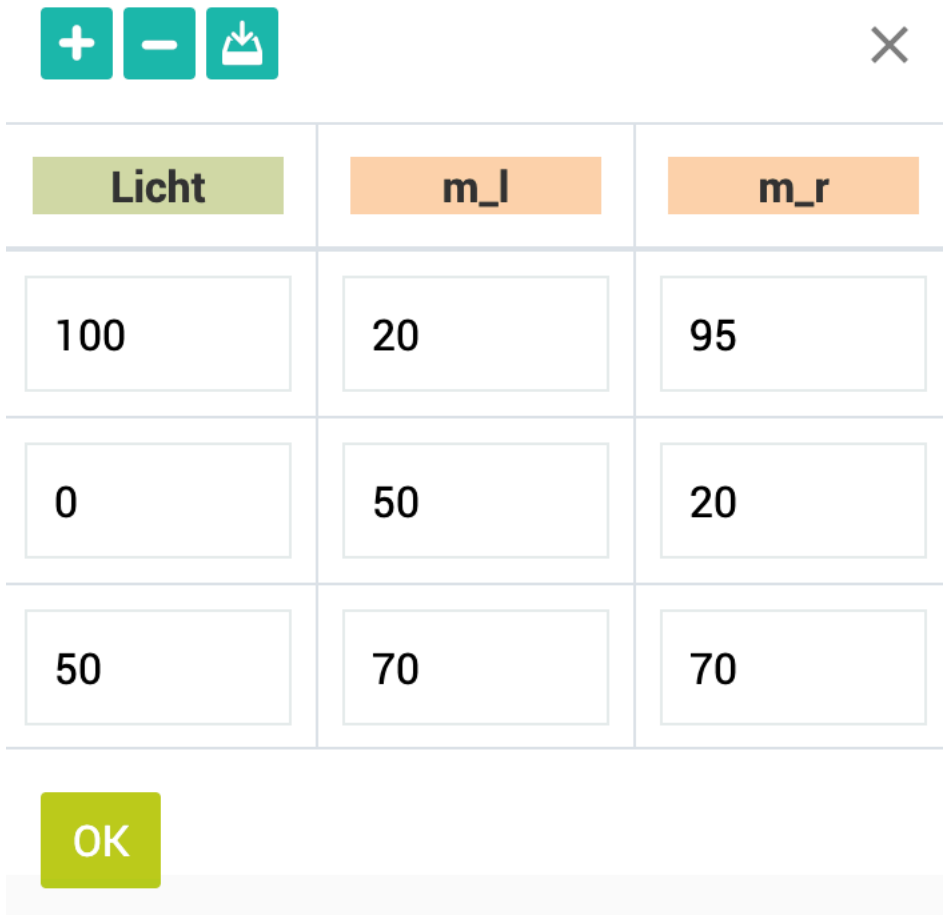


Fraunhofer IAIS

Neuronales Netz

Folgen einer schwarzen Linie

Tipp: Wenn der Roboter Schwierigkeiten hat, Kurven zu fahren, erhöhen Sie den Abstand zwischen den beiden Motorwerten für den Eingabewert weiß (100).



Fraunhofer IAIS

Kontakt

Thorsten Leimbach
Roberta-Zentrale@iais.fraunhofer.de

Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS
Schloss Birlinghoven 1
53757 Sankt Augustin

www.iais.fraunhofer.de