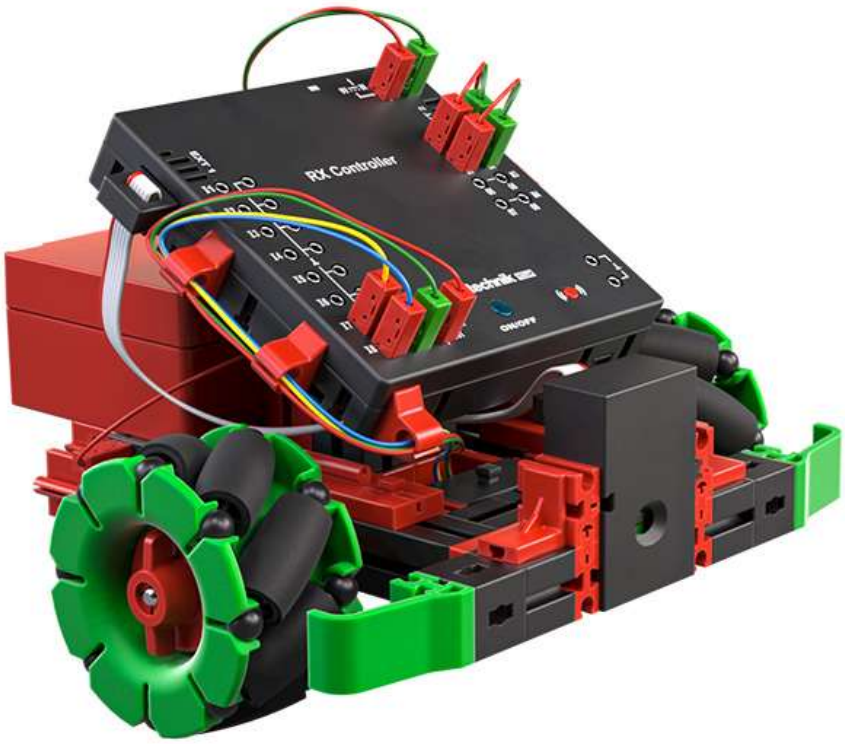
fischertechnik

AI Advance 2 wheeled robot mit KI

SMART Robots Max



Unser Roboter hat ein kleines Gehirn!

V2: Headline SemiBold

Dieser Roboter folgt keinen festen Regeln wie „wenn dies passiert, tue das“. Stattdessen hat er ein kleines künstliches Gehirn, ein sogenanntes neuronales Netz, das anhand von Beispielen lernt, genau wie du, wenn du Fahrradfahren lernst 🚲. Am Anfang kann er nichts…
Aber mit Training wird er immer besser.
Was ist ein neuronales Netz?
Stell dir vor, der Roboter hat:
👀 Augen → die Sensoren
🧠 Gehirn → das neuronale Netz
🦵 Beine → die Motoren
Die Augen sehen die Welt, das Gehirn entscheidet, was zu tun ist, und die Beine bewegen sich.

Was genau macht dieser Roboter?

Dieser Roboter:
Hat 2 IR-Track-Sensoren (schauen auf den Boden)
Hat 2 Motoren (links und rechts)
Möchte lernen, einer Linie zu folgen
Aber Achtung ⚠️
👉 Wir sagen ihm nicht, wie er es machen soll
👉 Wir zeigen es ihm anhand von Beispielen

Wie spricht der Roboter mit seinem Gehirn?
Eingaben (Inputs)
Das Gehirn des Roboters erwartet immer eine Liste von Werten.
In diesem Fall:

Linker Sensor → ein Wert
Rechter Sensor → ein Wert

Wie erstellt man die Liste?

Du erhältst die Blöcke im Menü „Data Structures“, wenn du „Learning level 3“ aktiviert hast. Zur Vereinfachung erstellen wir die Liste und weisen sie der Variablen „Inputs“ zu.

Die Sensoren liefern uns Werte von 0 oder 1. Neuronale Netze funktionieren besser mit Werten zwischen -1 und 1. Daher müssen wir nichts ändern.

Ausgänge (Outputs)

Das „Gehirn“ gibt eine weitere Liste zurück:

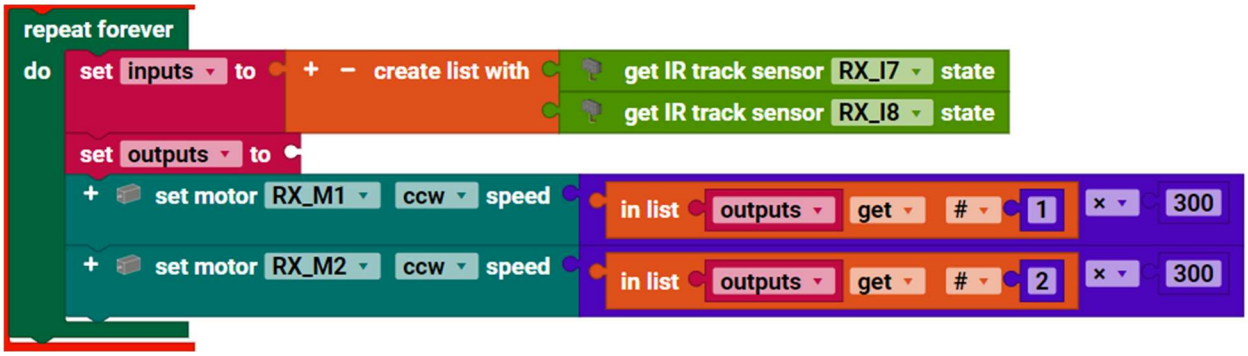
- Linker Motor → ein Wert
- Rechter Motor → ein Wert

Bevor wir mit der Konfiguration des Netzwerks beginnen, bereiten wir den Roboter weiter für unser „Gehirn“ vor.

Lege die Variable „outputs“ an. In dieser Liste speichert das neuronale Netz die Werte zur Steuerung der Motoren.

Wie bereits erwähnt, arbeitet das neuronale Netz am besten mit Werten zwischen -1 und 1. Wenn der Motor stehen bleiben soll, gibt das Netz 0 zurück, für maximale Geschwindigkeit Werte nahe 1, die auch größer als 1 sein können.

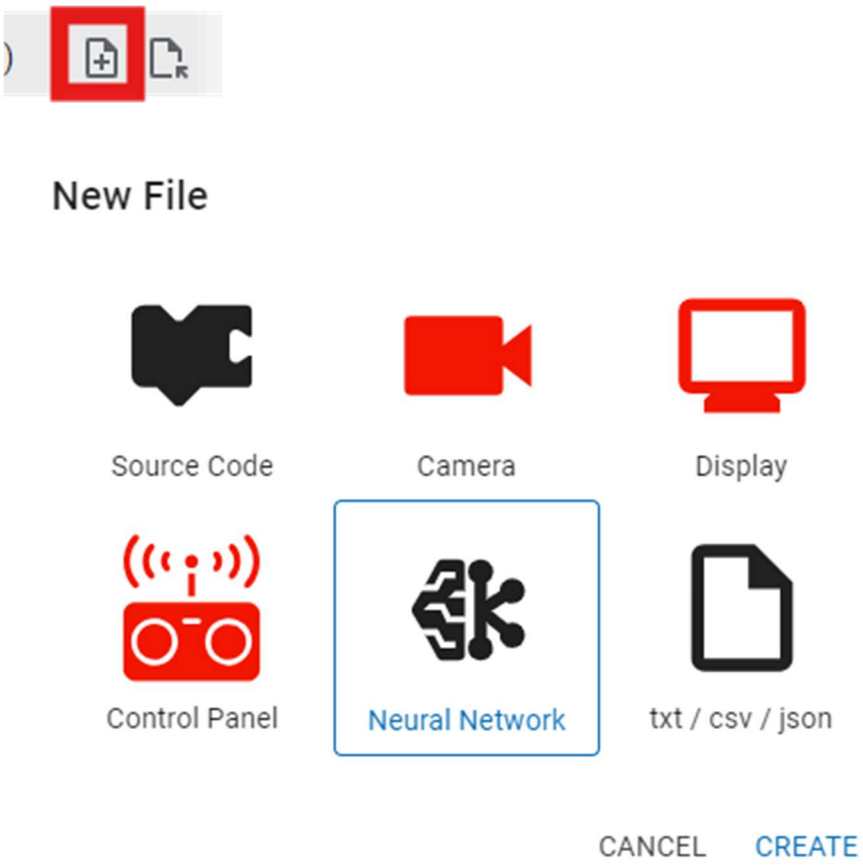
Da der Motor mit Werten von 0 bis 512 arbeitet, multiplizieren wir die vom neuronalen Netz gelieferten Werte einfach mit 300 (Werte zwischen 300 und 450 sind ideal), um die Motorgeschwindigkeit zu bestimmen. Wir nehmen nicht 512, falls das Netz einen Wert größer als 1 liefert.



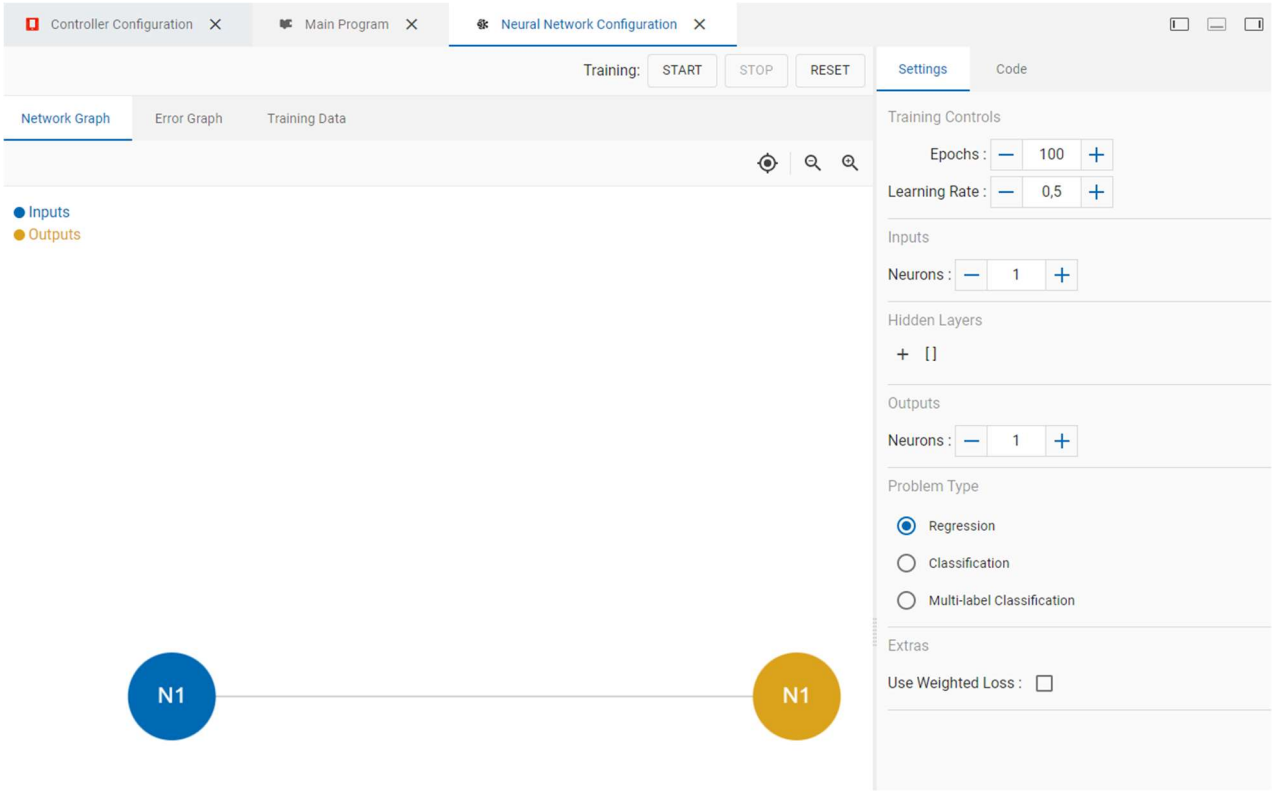
Wie du siehst, nehmen wir den ersten Wert der Liste für Motor 1 und den zweiten für Motor 2.

Das neuronale Netz erstellen.

Zunächst müssen wir das neuronale Netz erstellen und die Bibliothek „Neural Network“ hinzufügen.



Wir sollten folgendes Bild haben:



Wie viele Neuronen hat das „Gehirn“?

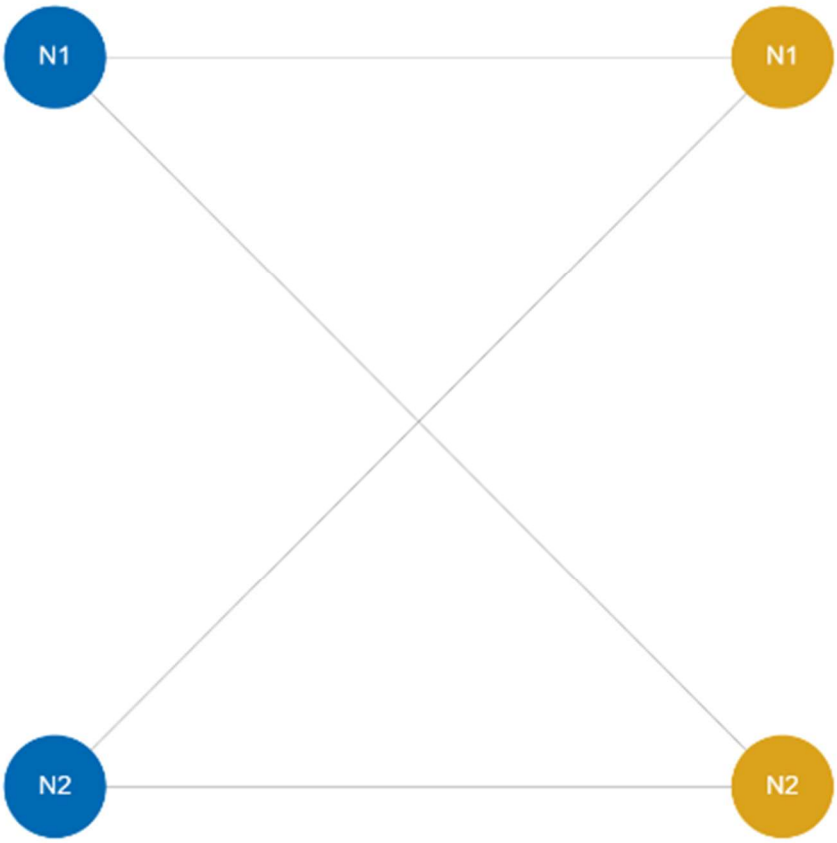
- Eingabeschicht (inputs)
  - 1 Neuron pro Sensor
  - 2 Sensoren → 2 Neuronen

- Ausgabeschicht
  - 1 Neuron pro Motor
  - 2 Motoren → 2 Neuronen

Das „Gehirn“ muss immer haben:

- so viele Eingabeneuronen wie Werte, die wir übergeben,
- so viele Ausgabeneuronen wie Dinge, die wir steuern möchten.

Füge also über das Menü an der Seite zwei Eingabeneuronen und zwei Neuronen in der Ausgabeschicht hinzu.



Das Gehirn des Roboters trainieren

Hier beginnt die Magie ✨
Trainieren bedeutet:
„Wenn du das siehst, dann sollst du das machen."
Wir geben dem Roboter ganz viele Beispiele,
und sein Gehirn passt sich Schritt für Schritt an.

⚙️ Wichtige Einstellungen (einfach erklärt)

🔁 Epochs

Wie oft der Roboter das Training wiederholt.

- Wenige → er lernt nur wenig
- Viele → er lernt besser (aber nur bis zu einem gewissen Punkt)

💡 Wie beim Üben eines Liedes:
Je öfter du übst, desto besser klingt es 🎵

🎚️ Learning Rate

Wie stark sich das Gehirn nach jedem Beispiel verändert.

- Zu groß (0,5) → der Roboter wird ganz durcheinander
- Gut ist zwischen 0,1 und 0,005

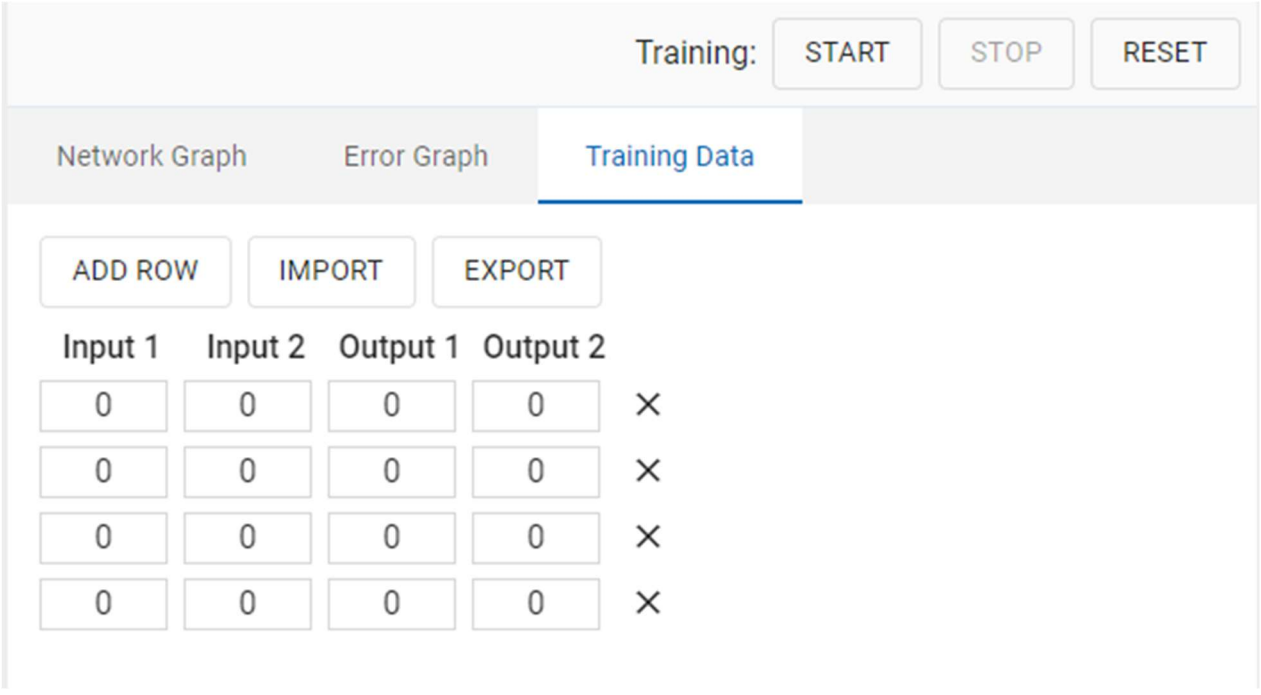
💡 Wie beim Lernen:

- Zu schnelle Änderungen → viele Fehler
- Langsame Änderungen → man lernt richtig gut

Trainingsbeispiele (Training Data)

Wie wir schon gesagt haben:
Eine neuronale Netz lernt durch **Beispiele**.
Deshalb müssen wir diese Beispiele eingeben.

In diesem Fall machen wir das **von Hand**.
Wir fügen **vier Zeilen** hinzu – das sind vier Beispiele.

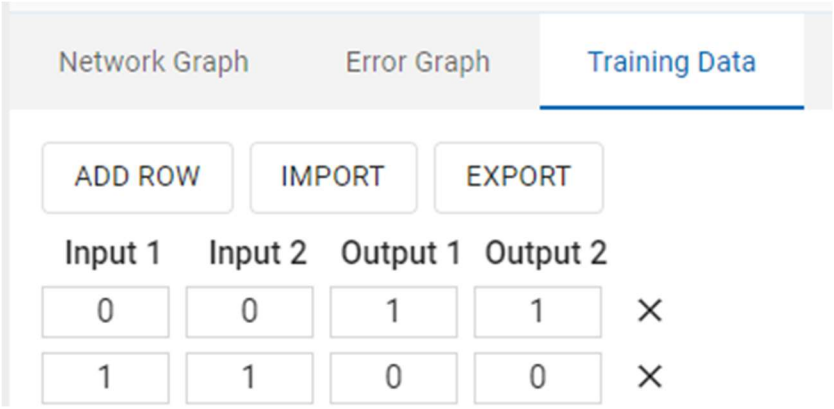


ie füllen wir die Werte aus?

Fall 1:
Beide Sensoren haben den Wert **0**.
➡️ Der Roboter ist perfekt auf der Linie
➡️ Er soll **mit maximaler Geschwindigkeit fahren**
(Das ist ein Wert nahe bei **1**)

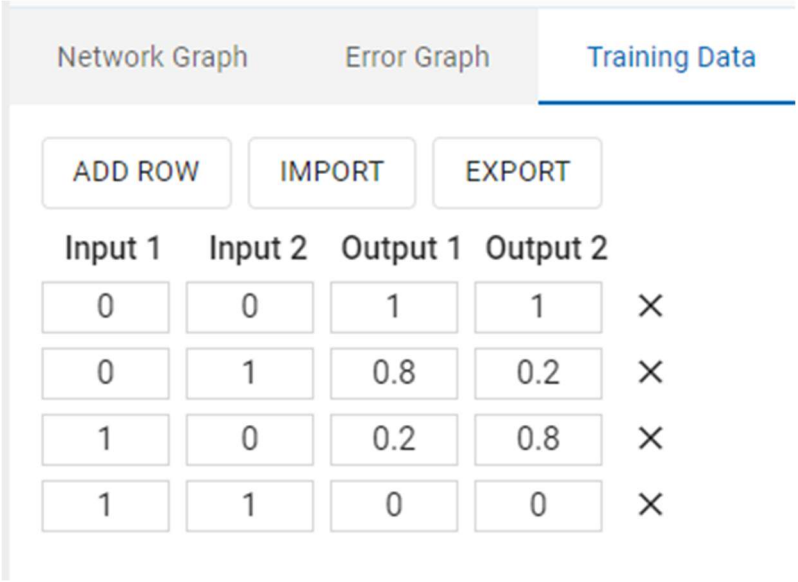
Input 1	Input 2	Output 1	Output 2	
0	0	1	1	✕

Fall 2:
Beide Sensoren haben den Wert **1**.
➡️ Der Roboter hat die Linie verloren
➡️ Der Roboter soll **stehen bleiben**



Fall 3 und 4:
Ein Sensor ist 0, der andere 1.
➡️ Der Roboter verliert gerade die Linie
➡️ Er muss lenken

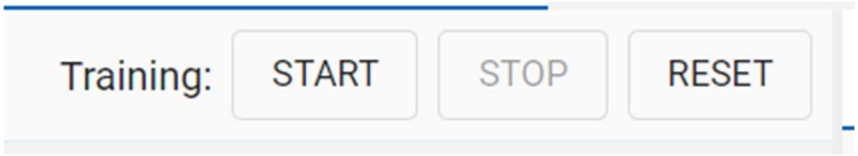
Wie?
👉 Ein Motor dreht sich schneller als der andere.
Hier siehst du ein mögliches Beispiel:



Wenn alle Daten da sind, können wir trainieren.
Jetzt lernt die neuronale Netz wirklich 🎉

Das Training starten

Nachdem wir alle Daten eingegeben haben,
starten wir das Training mit **Start**.



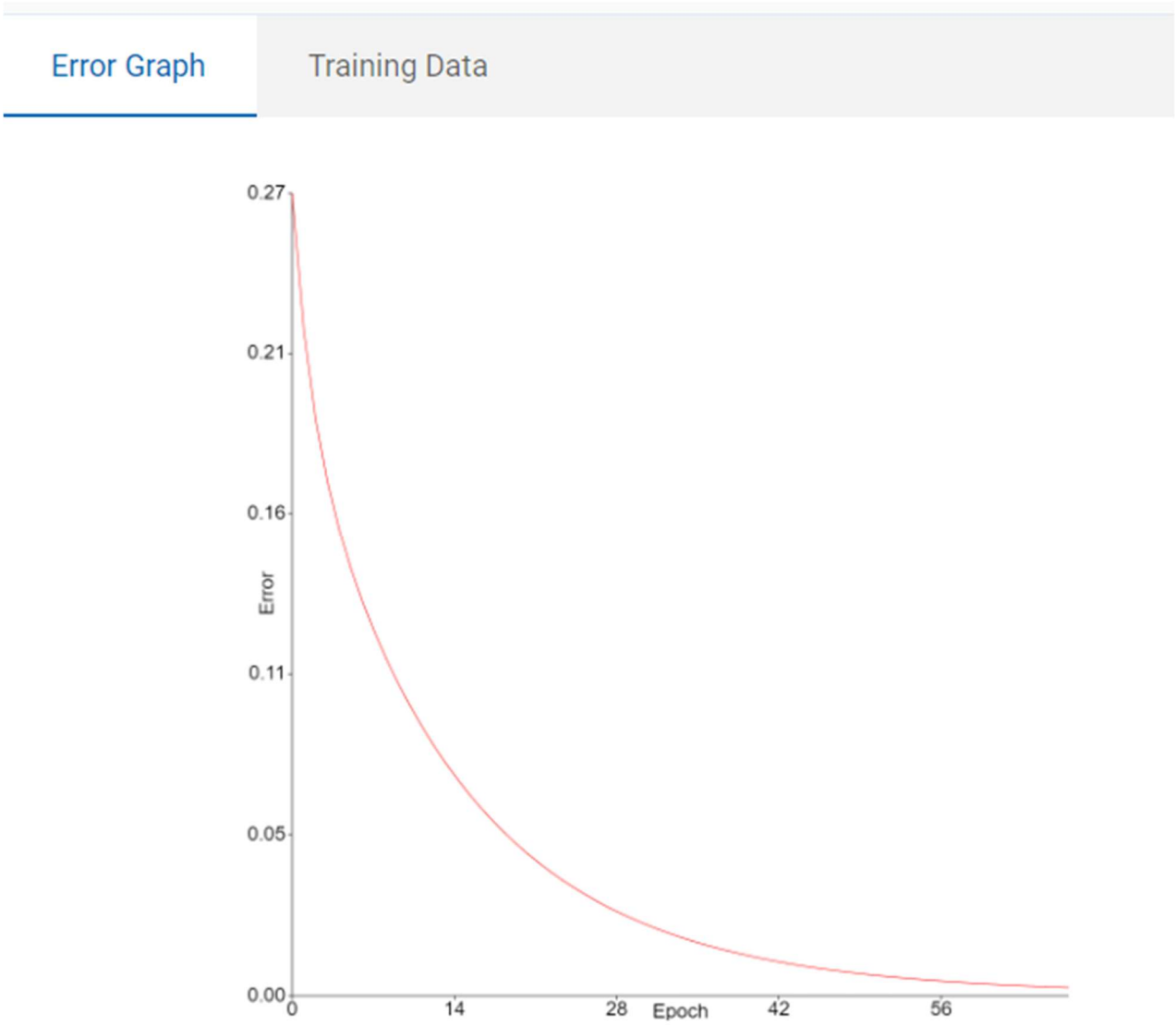
Fehler (Error)

Der Fehler zeigt, **wie oft sich der Roboter irrt**.

- Großer Fehler → er hat noch nicht gelernt
- Kleiner Fehler → er macht es schon ziemlich gut

⚠️ **Wichtig:**
Ein Fehler von **0** ist nicht immer nötig oder möglich.
Wichtig ist, dass der Roboter **in der echten Welt gut funktioniert**.

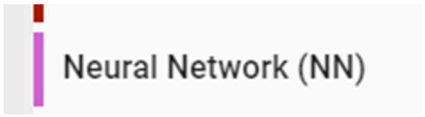
Eine Grafik wie diese zeigt,
dass die neuronale Netz **sehr gut gelernt hat** 👍



Das Gehirn einsetzen

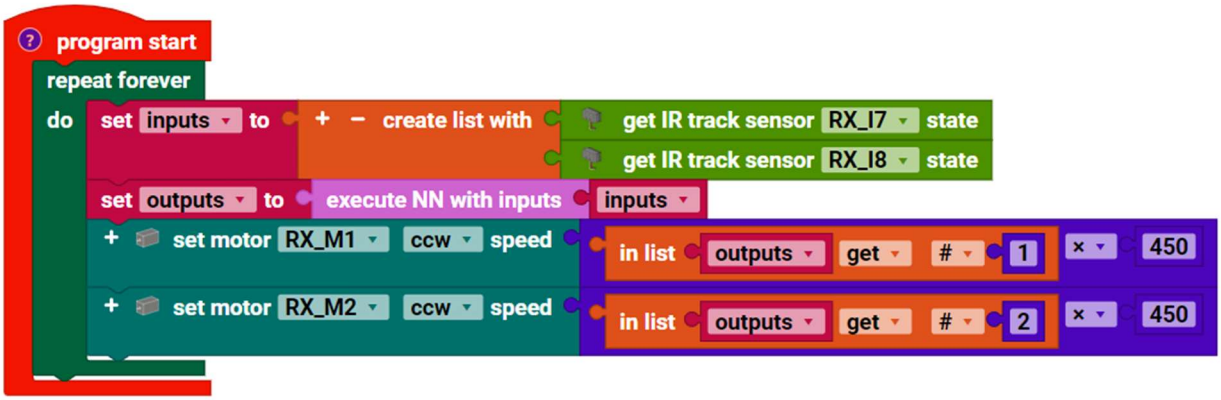
Jetzt ist die neuronale Netz fertig trainiert.
Nun müssen wir sie nur noch **richtig im Programm benutzen**.

Gehe ins Menü
und ziehe den Block **„execute NN with inputs“** ins Programm.



- Inputs → die Sensorwerte
- Outputs → die Motorwerte

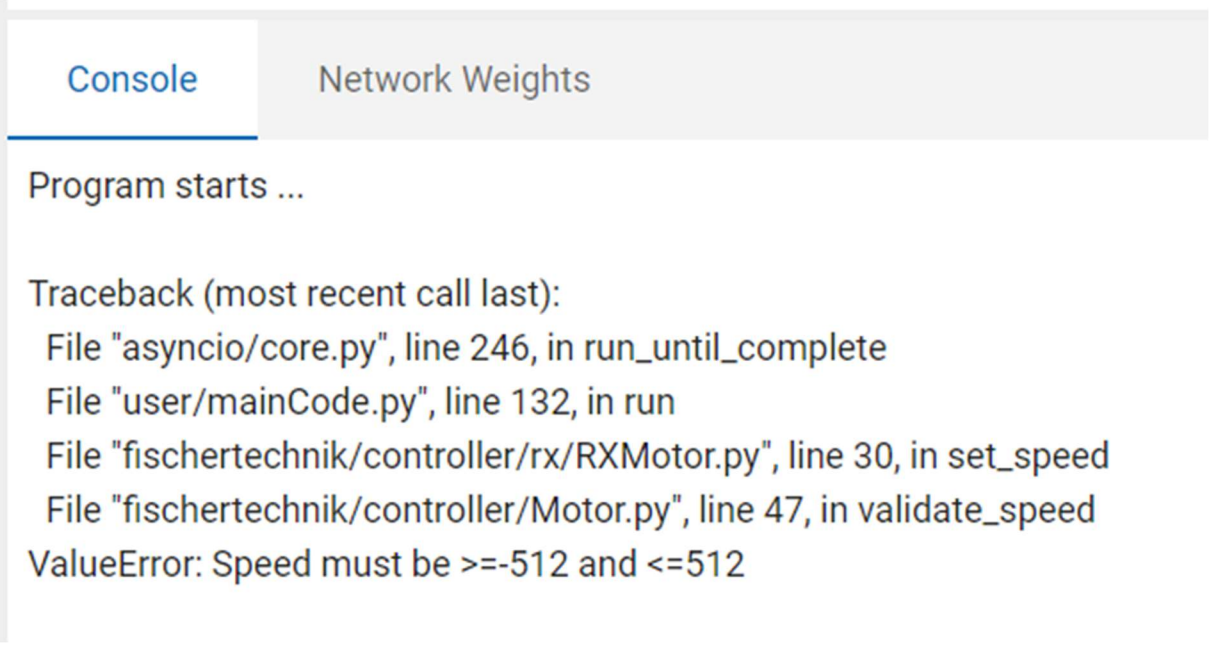
So sollte das Programm aussehen:



Möglicher Fehler

Neuronale Netze sind mathematische Funktionen.
Manchmal ist die Ausgabe nicht ganz so, wie wir es erwarten.

Dann kann so ein Fehler auftreten.



Was kannst du tun?

- Das Training verbessern
- Oder die Motorwerte kleiner machen

Merke dir das

Wenn wir mit

- Robotern
- neuronalen Netzen
- künstlicher Intelligenz

arbeiten, dann denkt dort **keine Person**.

Es gibt nur:
➡️ Mathematik
➡️ Funktionen
➡️ Beispiele

🎮 **Jetzt bist du dran!**

Eine Idee für dich:
Was wäre, wenn der Roboter rückwärts fährt,
wenn er die Linie verliert?

👉 **Ändere diese Zeile,**
mache Reset
und trainiere den Roboter neu.

Beobachte die Fehler-Grafik.
Starte danach das Programm ohne den Code zu ändern.

Schau genau hin:
Der Roboter verhält sich jetzt **anders** 😄

Das komplette Programm findest du bei den Beispielen:
„AI_Advanced_2_wheeled_robot_1“

Aber bleib nicht bei diesem Beispiel!
🎲 Spiele weiter und probiere neue Trainingswerte aus.

Hindernisse erkennen!

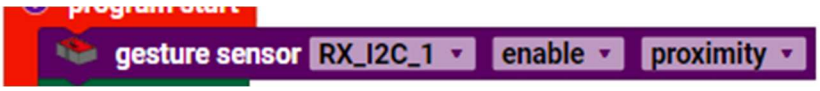
Jetzt nutzen wir den **RGB-Sensor**,
um Hindernisse zu erkennen,
und lassen die neuronale Netz **selbst reagieren**.

Hier siehst du,
warum neuronale Netze so stark sind
und warum sie die Welt der KI verändert haben 🌍

Los geht's!

Sensor einstellen

Stelle den Sensor so ein,
dass er **Abstände misst**.



Mehr Infos für die neuronale Netz

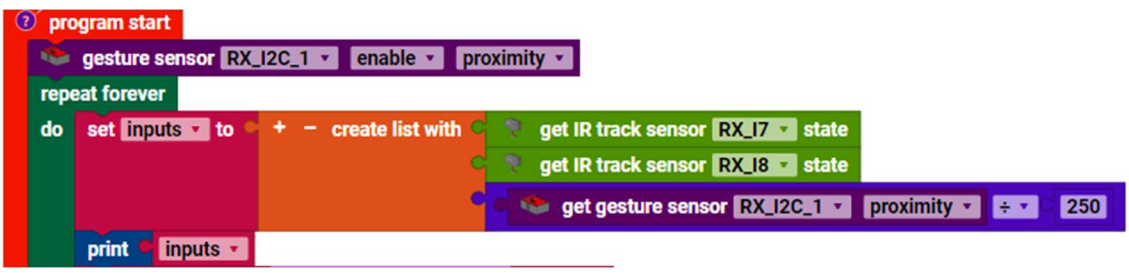
Jetzt geben wir der neuronalen Netz **mehr Informationen**.

Wir fügen den Sensorwert zur Liste hinzu.

Aber Achtung:
Der Sensor liefert Werte bis **250**.

Die neuronale Netz arbeitet besser mit Werten **kleiner als 1**.
👉 Deshalb teilen wir den Sensorwert einfach durch **250**.

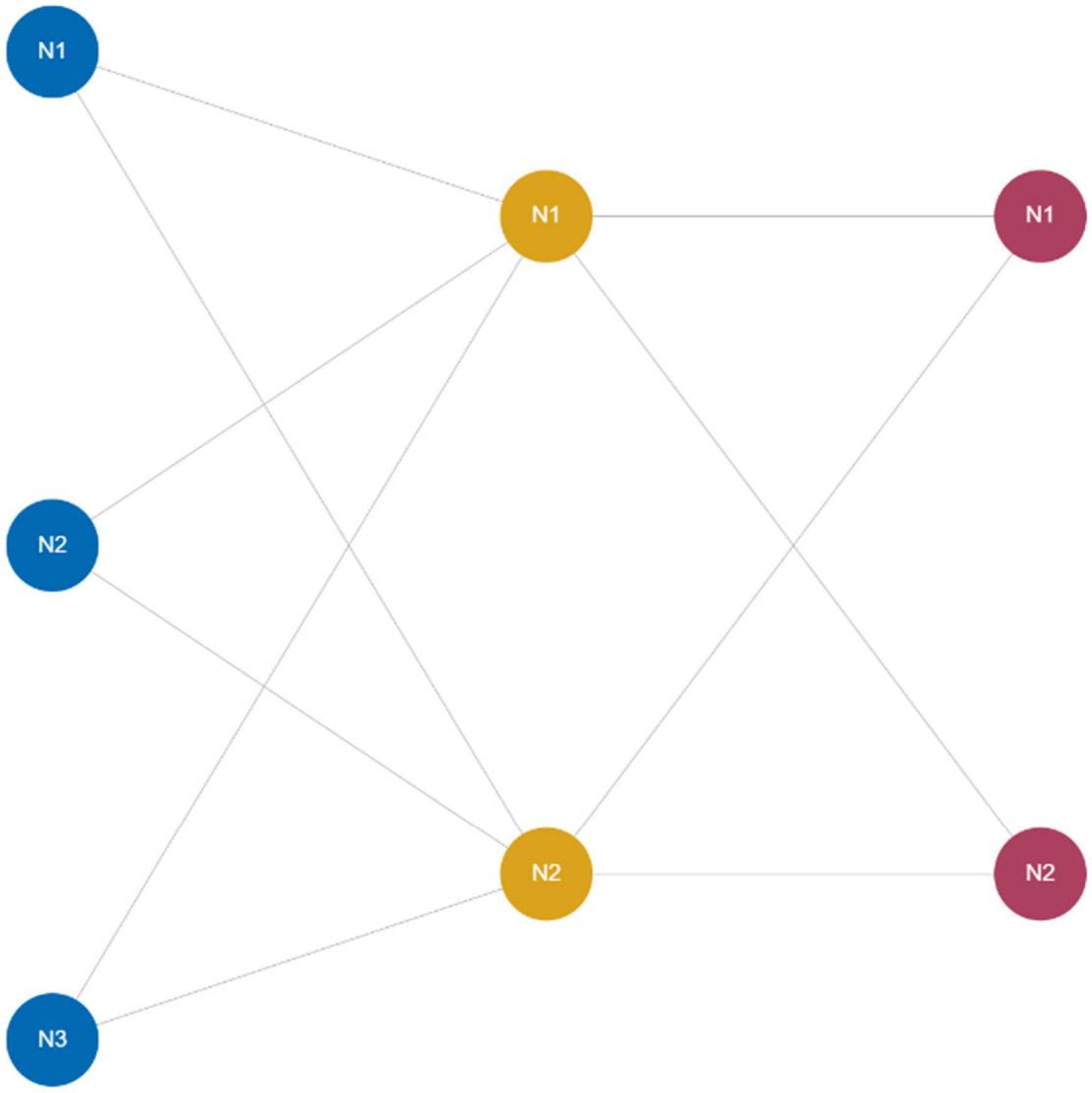
.



Die neuronale Netz anpassen

Damit der Sensor gelesen werden kann,
fügen wir eine neue Eingangsneurone hinzu.

💡 Tipp:
Erstelle zusätzlich eine versteckte Schicht
mit ein paar Neuronen.



Trainingsdaten erweitern

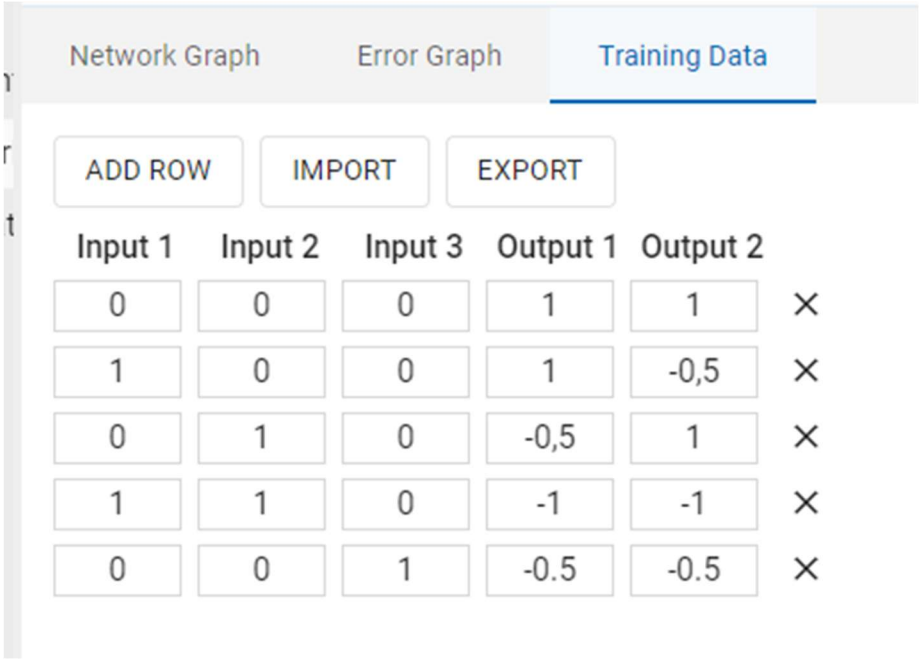
Jetzt gibt es eine **neue Spalte**
für die dritte Eingangsneurone.
Am Anfang steht überall **0**.
Das ist gut so!
0 bedeutet: **kein Hindernis**

➡️ Der Roboter soll sich so verhalten wie vorher.

Jetzt fügen wir **eine neue Zeile** hinzu:

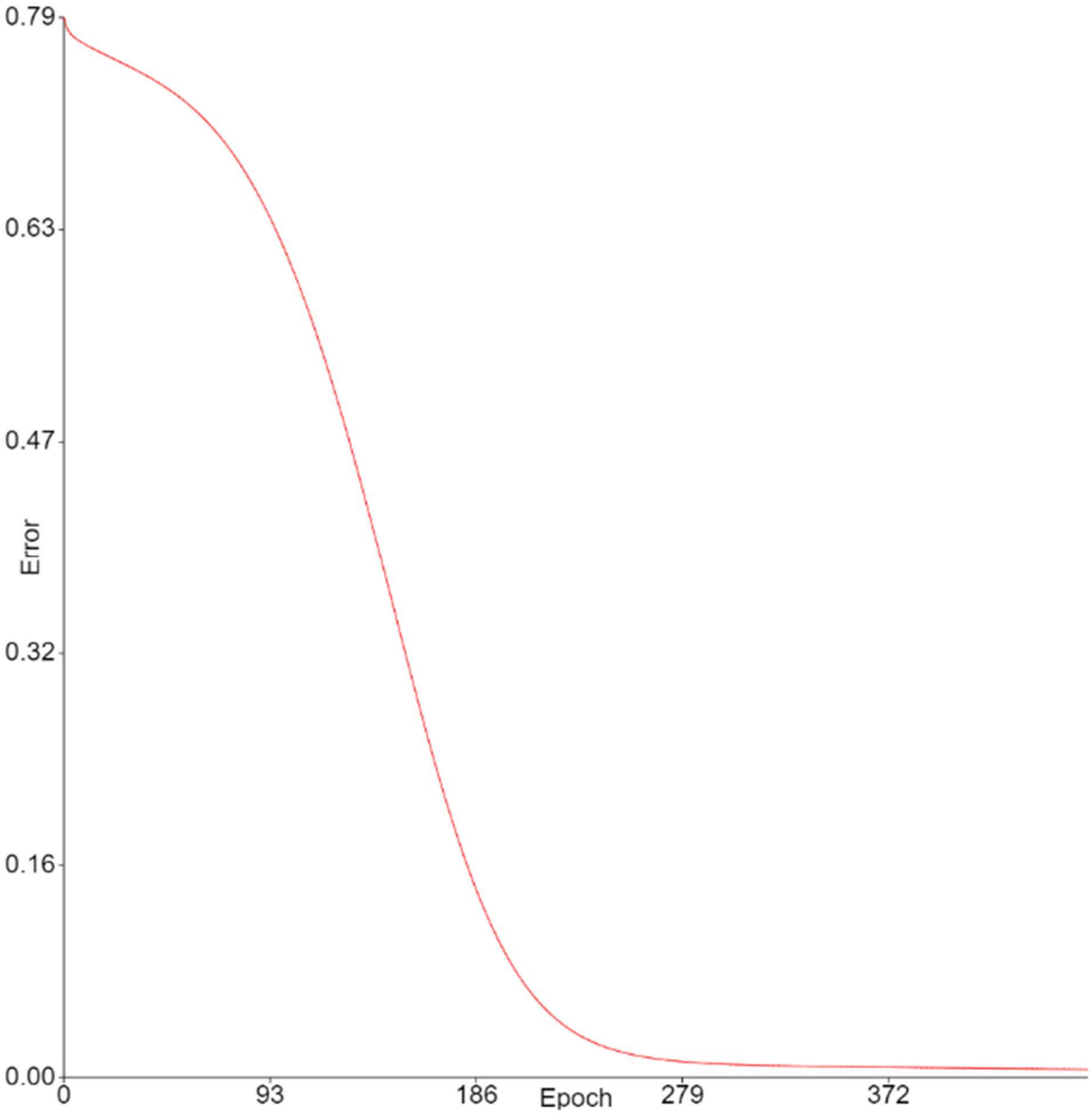
- Roboter ist auf der Linie
- Hindernis erkannt (N3 = 1)
  - ➡️ Roboter fährt **rückwärts mit halber Geschwindigkeit**

.



Trainieren & testen

Jetzt kannst du wieder trainieren.
Spiele mit **Epochs** und **Learning Rate**,
bis du eine Grafik mit **kleinem Fehler** bekommst.



Starte den Roboter und halte deine Hand vor den Sensor
👉 so simulierst du ein Hindernis ✋

Fazit

Schau dir diesen letzten Fall genau an:
Wir haben **nur ein neues Beispiel** hinzugefügt, und trotzdem trifft der Roboter **richtige Entscheidungen** – auch beim Drehen oder Ausweichen.

Wenn wir alles **normal programmieren** müssten,bräuchten wir ganz viele extra Regeln
für jede Situation. Die neuronale Netz braucht das nicht.

Das ist der große Vorteil von neuronalen Netzen:

Sie müssen **nicht alle Fälle vorher kennen**,
um richtig zu reagieren.

Deshalb kann eine KI heute, nachdem sie viele Katzenbilder gesehen hat,
auch eine neue Katze erkennen 😺

Was du hier gemacht hast, ist im Kleinen genau das, was große künstliche Intelligenzen heute tun.

Sie bestehen aus riesigen neuronalen Netzen mit Millionen von Daten –
aber die **Grundidee ist genau dieselbe**.

Zum Schluss noch wichtig:
Wir haben hier den **Problemtyp Regression** benutzt.

Diese Art ist perfekt, wenn die Ausgabeneuronen **verschiedene Werte** haben sollen.

👏 Super gemacht – jetzt weiter ausprobieren und entdecken!