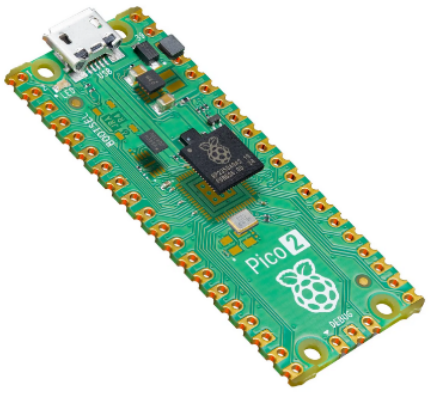
KI-ENNA (2.0)

(E)in (N)euronales (N)etz...

KI-ENNA ermöglicht **ressourceneffiziente Deep Learning Modelle** auf Microcontrollern als Embedded Systems. Hierzu können nach dem Pretraining mit TensorFlow und Keras die Parameter in MicroPython für ein **Offline- und Echtzeit-Monitoring** auf KI-ENNA überführt werden.

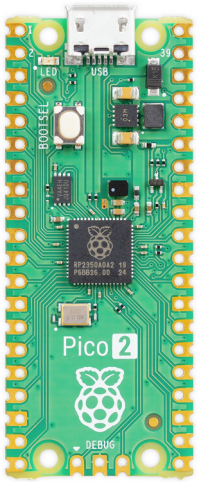
...zum (A)usprobieren

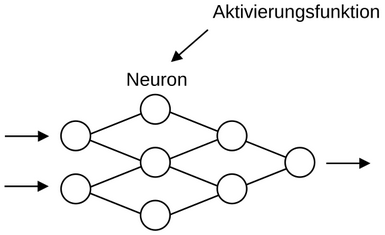
KI-ENNA ist auch ein **didaktisches Tool** für Hochschulen, Berufsschulen und allgemeinbildende Schulen. Ziel ist die einfache Vermittlung der **Funktionsweise von Neuronalen Netzen** (Aktivierungsfunktionen, Parameter, etc.) mit klassischen **Data Science** Beispielen.



Mehr Informationen und kostenloser Download:

KINRW Lernende Systeme – Die Plattform für Künstliche Intelligenz GitHub





Einführung in Neuronale Netze

Bereits **mit zwei funktionsfähigen Neuronalen Netzen** ausgestattet lässt sich KI-ENNA sowohl für die Praxis als auch für Lehr- und Lernsettings **flexibel anpassen**. Hier geht es um die Neuronen, Schichten und Funktionen von Neuronalen Netzen.

Mathematische Grundlagen

KI-ENNA führt anschaulich an die mathematischen Grundlagen von Neuronalen Netzen heran und erklärt **mit praktischen Beispielen** deren Funktionsweise. Sigmoid, Tanh, ReLU, Leaky ReLU und Softmax lassen sich direkt als **Funktionen ausprobieren**.

$$LeakyReLU(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

GitHub


```
# Leaky ReLU
def leaky_relu(x, alpha=0.01):
    p = []
    for i in range(len(x)):
        if x[i] > 0:
            p.append(x[i])
        else:
            p.append(alpha * x[i])
    return p
```

Programmieren mit MicroPython

Mit KI-ENNA können die mathematischen Grundlagen einfach in MicroPython programmiert und direkt für die **Verwendung auf einem Microcontroller** umgesetzt werden. So können bspw. **eine Matrix transponiert und Funktionen programmiert** werden.