Advancements in teaching AI in medium level education

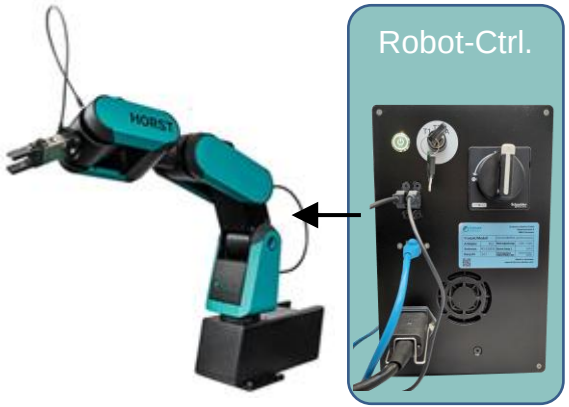
R.Müller[1], M.Abrell[1], Ch.Bildhauer-Buggle[1], Th.Schiepp[1]

1: Furtwangen Hochschule University, Faculty of Engineering and Technology
Campus Schwenningen & Research Center Rottweil, Germany
(contact: thomas.schiepp@hs-furtwangen.de)

18.6.2026

Our previous efforts

In our presentation in 2024 we already introduced our combined AI-robot setup:



industrial Robot
(Fruitcore Horst600)

Robot-Ctrl.

dig.Trigger

status flag

Camera

image

8x DI

16x DO

Dig. Interface

USB

DIO-USB-interface
(Deditec)

AI-PC

CPU-process

ai-handling software

image

Gra.Ca.

AI-Module

results table

AI-optimized PC
(i-mation)

COGNEX VISIONPRO

AI handling & C# - interface

COGNEX VIDI

Locate Analyze Read Classify

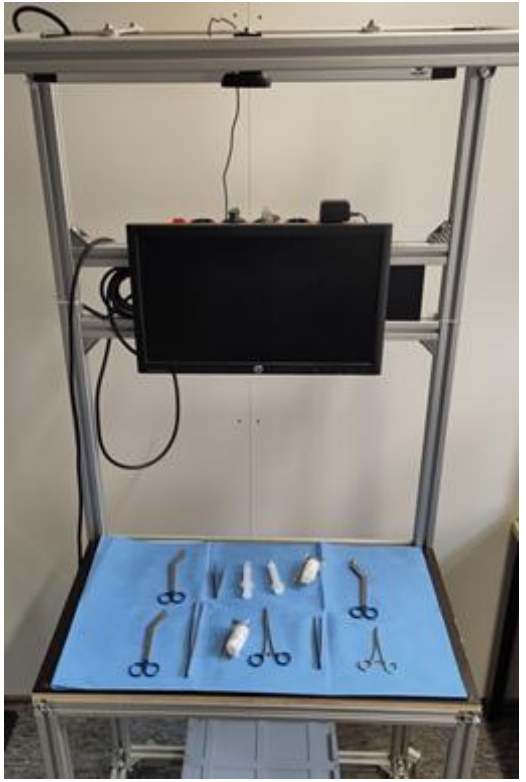
AI trainer GUI

high-res-camera + AI software
(CognexVision)

Our lab setups

We installed 2 additional hardware-setups:

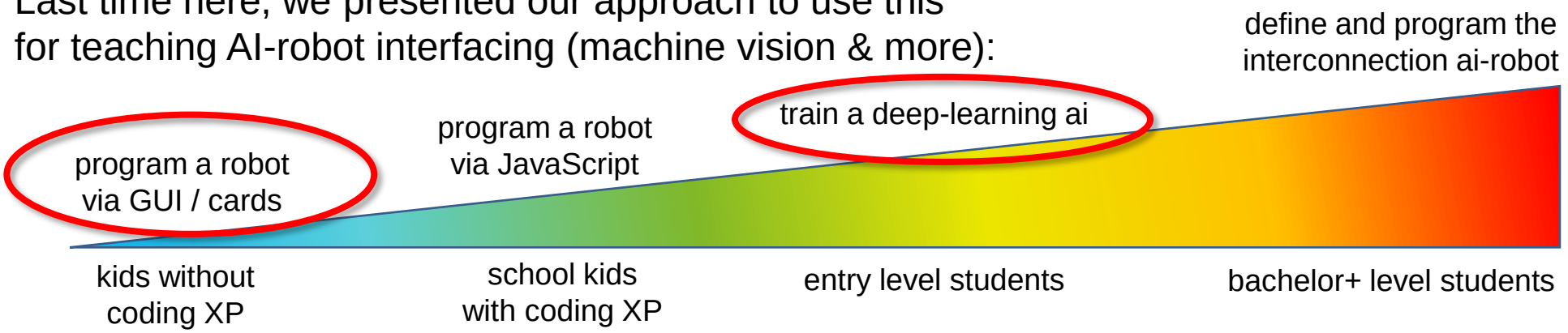
a car-trunc transportable "YOLO" stand; webcam-compatible CS-mount camera + Jetson-nano (AI-enhanced micro-computer) with Linux and YOLO (details by Markus Abrell)



an extension of our robot setup by another standard desktop PC with a RTX4060 GraphicsCard, CUDA, Anaconda, Python and PyTorch

Our course as of today

Last time here, we presented our approach to use this for teaching AI-robot interfacing (machine vision & more):



The course focuses on AI-modeling and to a smaller extent introduction to robot programming with the primary goal to teach machine-vision for robot guidance

The accompanying lecture is comprised of 4 main theme areas

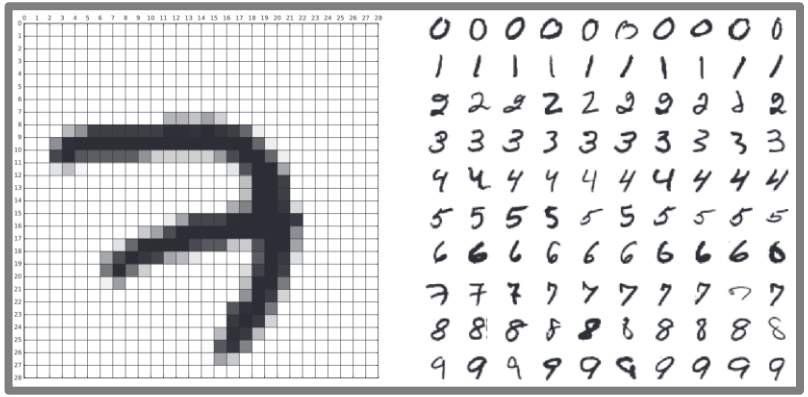
- 3 introductory lectures (90 min.): applications, theory and coding of AI, hardware
- 2 lectures and 1 lab exercise with YOLO: how to use an open-source model
- 2 lectures and 1 lab exercise with a robot and a commercial AI system: how to program a robot and how to use the (ready implemented) Cognex-Vidi system
- 1 lecture and a home-assignment: use a Pytorch template to solve the MNIST-problem

… what's the MNIST inference task…

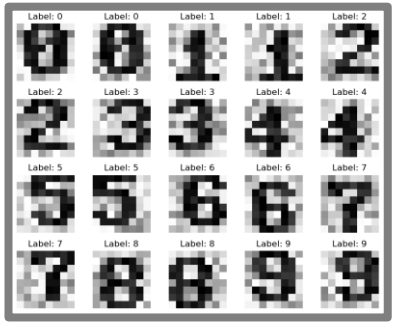
Standard exercise for every **code** programmer is the famous "Hello World" program

For **AI networks**, "hello-world" exercise is the recognition of the MNIST / EMNIST database for text recognition (by Yann Lecun at MIT) ; numerous videos, online help, tutorials and publications online

The classical MNIST data set: a collection of 70,000 images, 28x28 pixels of handwritten number symbols, ready labelled and prepared in a Python data format
The assignment: create a simple Neural Net to identify the handwriting (classification)



We created a new 8x8 pixels dataset, starting from the classical C64 characters:



This dataset is much smaller (~1000) and requires also a much smaller AI network for success and runs on any laptop

Our lecture tasks

The 3 tasks: **A)** configure YOLO to recognize any chosen objects (e.g. medical instruments)

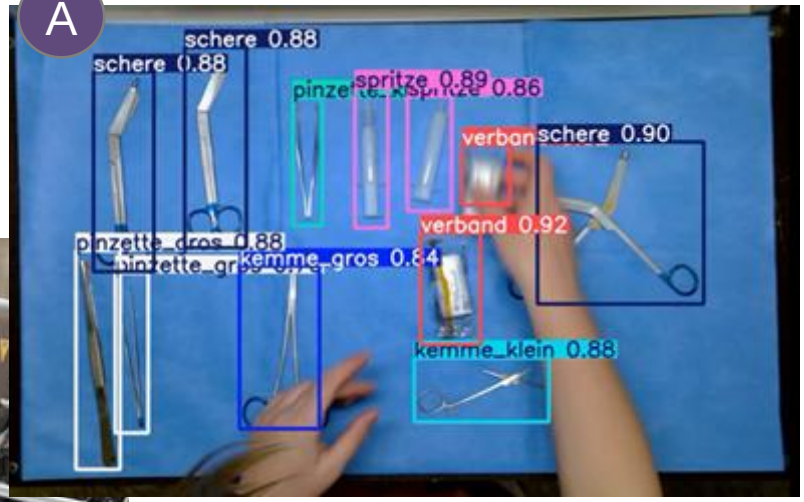
B) Recognize color and type of a target and let the robot sort them by color to a rail or a bin (a lot of code for the interface provided)

C) Solve the MNIST inference task with Python and Pytorch

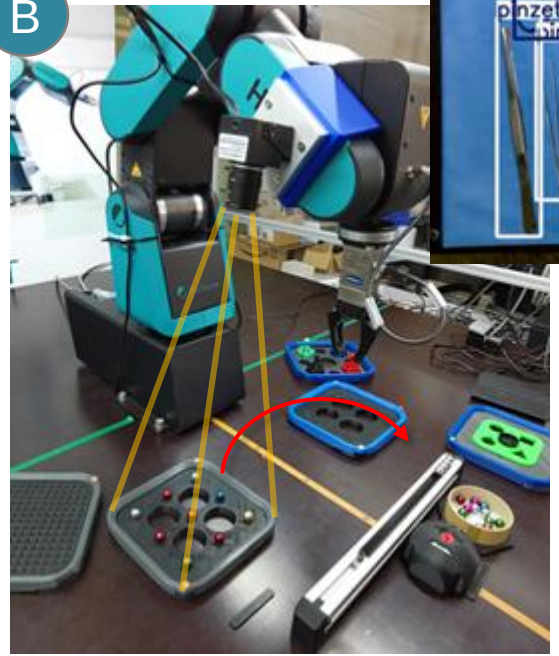
Our student receive a template modified for poor performance + the "C64-dataset"

the assignment is to modify the template to enable it to read the new input and achieve >95% inference sucess rate.

A



B



C

PyTorch

Our experience with didactic success

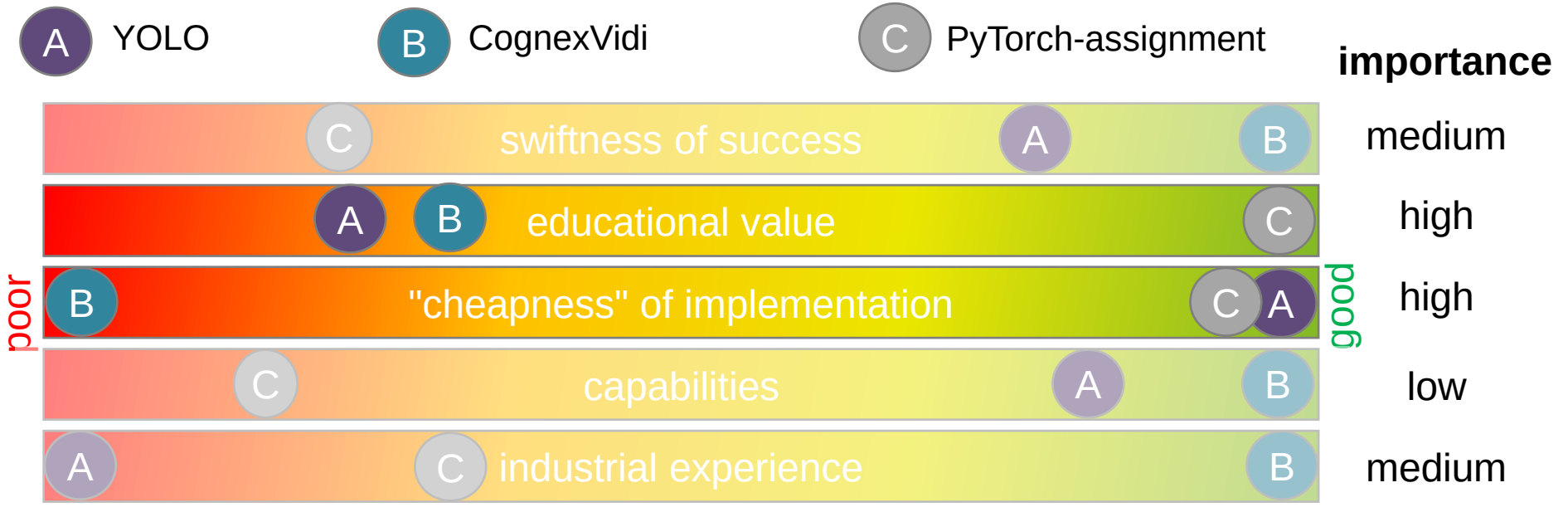
Acceptance amongst students very positive, fully booked attendee lists, despite being a voluntary course; offered in both german and englisch language versions

Analyzing the didactic success, a rather extreme contrast showed with no clear winner:

A YOLO

B CognexVidi

C PyTorch-assignment



greatest educational value AND easiest implementation : the programming task

salute to Yann Lecun!

Lessons and follow-up approach

However, the course suffers 2 bottlenecks: only 1 AI model/device for controlling the robot and only 1 robot to interface. Number of course-participants was limited to 8; need to expand.
Also, we would like to offer this material and course concept to other faculties / institutions (a bit like the MNIST exercise, but for CNN's)

We have a few industry robots with interfaces, but where to get an AI model with more control:

Path A: (our initial plan) use YOLO (by ultralytics.com) : a popular, open source AI software very powerful and highly developed; if interfacing or integration into a controlling software (preferably in python) achievable, this might suceed

ultralytics

Path B: use the experience with PyTorch and develop our own AI network

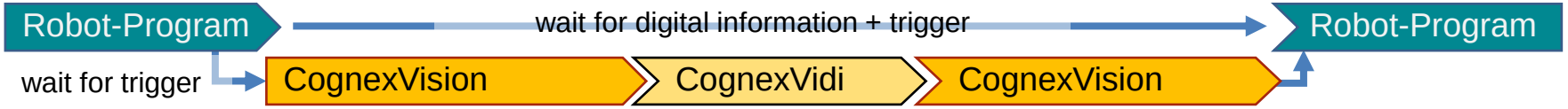
Our assignment with the MNIST database showed encouraging success and PyTorch is used in industry circles also

Primary question: How much effort would it be, to program an AI model in the Pytorch framework plus all interfaces to replicate the function of the CognexVision and Vidi software at least for finding and identifying colored steel balls in a simple, steady environment?

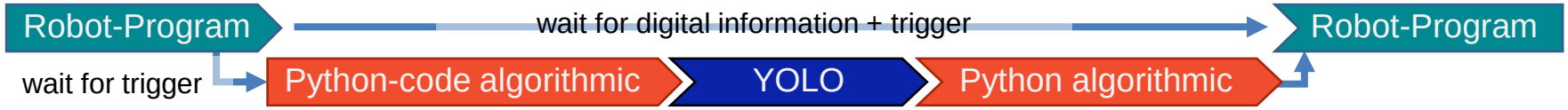
Running a robot and AI together

Interfacing the AI-model with a robot requires an intermediary system; the commercial system came with such a structure right away: the AI model trained by CognexVidi is a subserviant module of VisionPro.

The control is handed from the RobotProgram to the AI-software and back



To replicate this approach, replacing CognexVidi with YOLO appeared straightforward:



But attempts to use YOLO as a module inside a controlling python program failed
(it **IS** possible, but far too complex to be integrated in a 1 semester course)

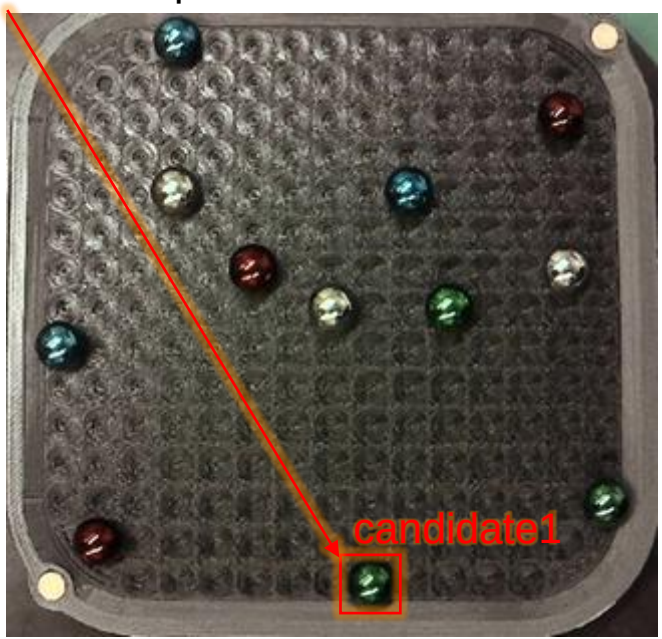
Which leaves path B….

A very brief fly-over of ConvolutionalNeuralNetworks

The task of the AI module:

Which class (in this case color) of balls is to be found where? make a list of positions and colors

1st AI input:



These are 2 questions:

a) which class
b) where in the image

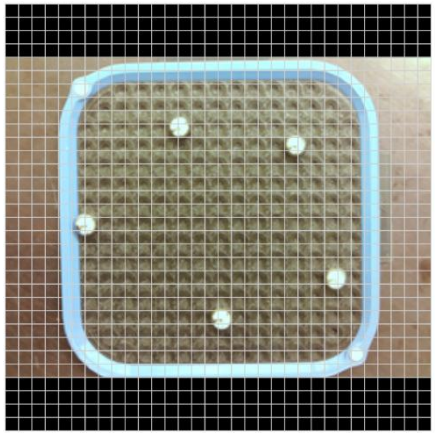
The MNIST-task only asked for the class, the position was square in the center

The standard way for modern AI's is to use 2 AI-networks, a first to find "candidates" and a second (that gets fed candidat coordinates along with the image) to classify the "object" (pattern in the image)

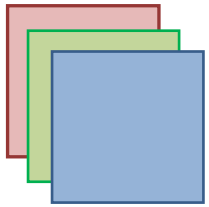
2nd AI input:

A very brief fly-over of ConvolutionalNeuralNetworks

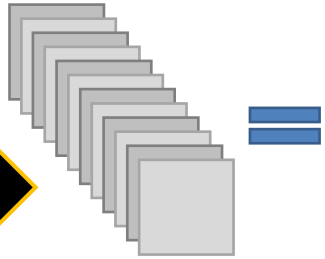
A **Convolutional Neural Network** takes a 3-channel input image (3 colors)….

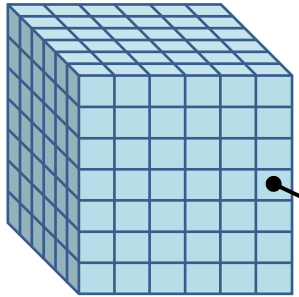


each pixel is a vector of 3 numbers (RGB):



convolution





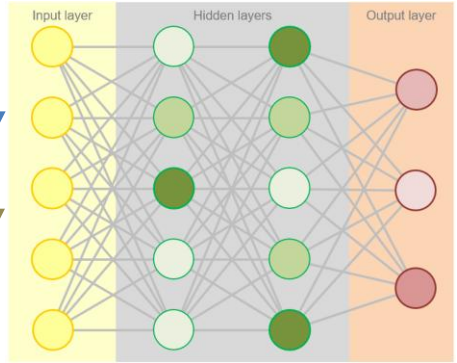
each cell contains a number, so each pixel contains a vector of numbers

$$\begin{pmatrix} no_1 \\ \vdots \\ no_{\#channels} \end{pmatrix}$$

… and generates a "feature map" : an "image" of vectors

where

A **Feedforward Neural Net** (like for the MNIST task) turns arrays into vectors and outputs a probability vector



inference

probability for class 1

probability for class 2

probability for class 3

what

Structuring a CNN with PyTorch

With PyTorch and Python, setting the "**parameters**" of an AI-network is actually rather short:

```
class ConvNet4x4(nn.Module):
    def __init__(self, num_channels_out):  # num_channels_out = size of output vector
        super().__init__()
        self.conv1 = nn.Conv2d( 3, 16, 3, padding=1)
        self.conv2 = nn.Conv2d(16, 32, 3, padding=1)
        self.conv3 = nn.Conv2d(32, 64, 3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.output_conv = nn.Conv2d(64, num_channels_out, 1)
```

# of incoming channels

# of outgoing channels

defines a pooling added to each layer

These few lines define a 3 Layer-CNN!

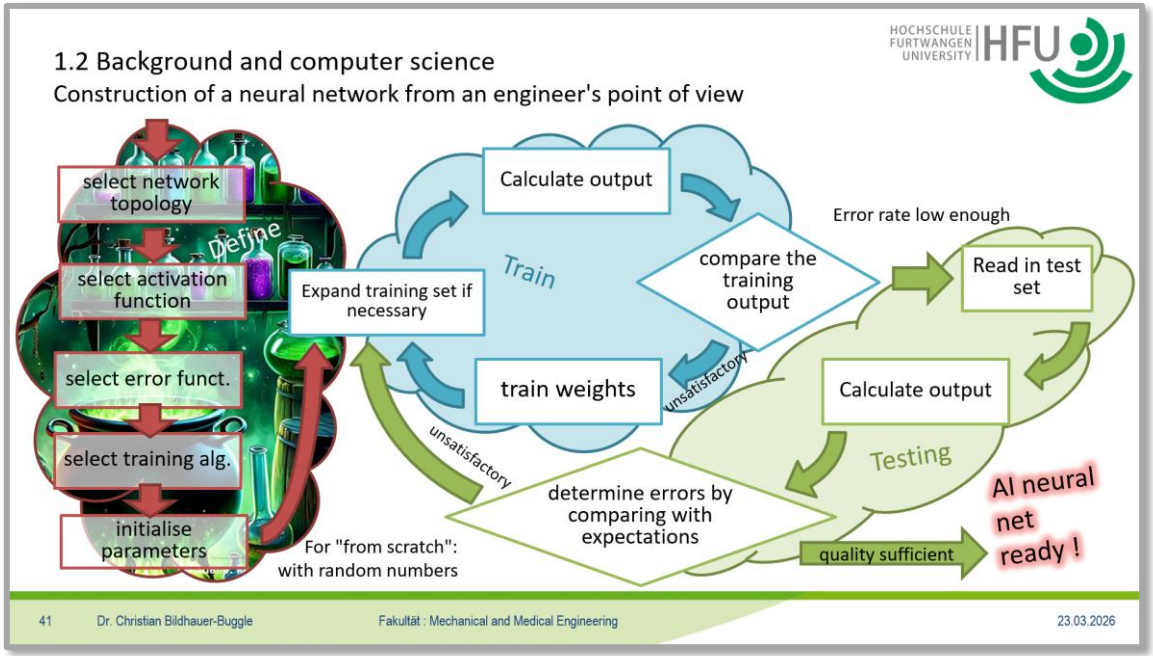
Next to the network itself, the way how it is trained must be specified by choosing the so-called "**hyperparameters**"

The 2 networks are combined by simply using the output of the first as input to the second; one can merge them into 1, but we chose not to.

Challenge: get this CNN as small as possible

Structuring a CNN with PyTorch

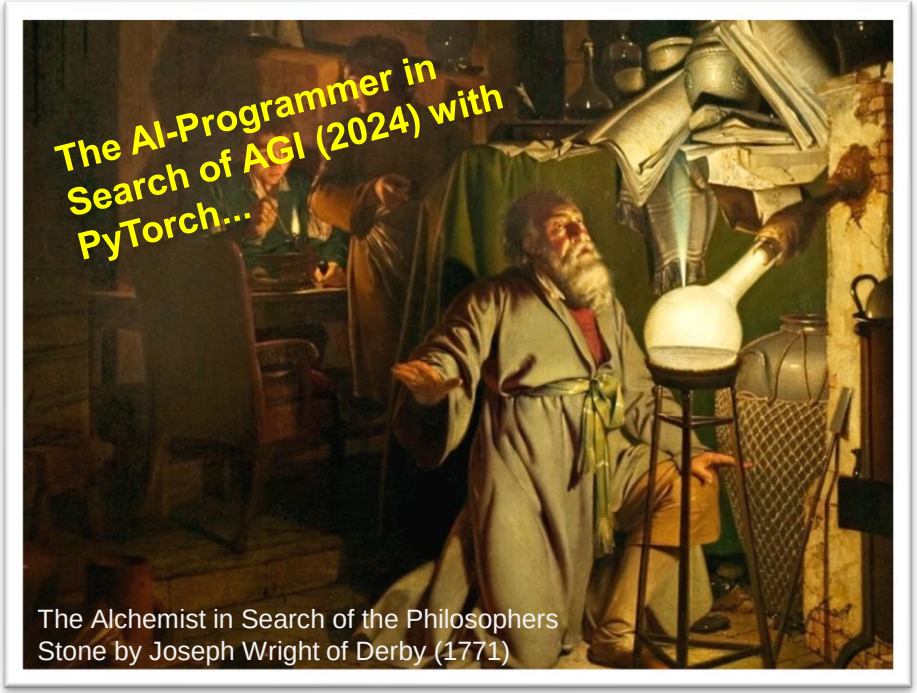
How to "find" an appropriate AI network:
an actual, serious slide from my lecture:



Grand-sum-total: it took 1 student (Ramona Müller, incl. bachelor thesis) about 1 year and 400 working hours to create the entire software suite, a set of 6 files with 1600 lines of code sum-total

AI development feels actually like alchemy with endless cycles of try-error-and-optimize, consuming lots of work-time (fast PC-hardware helps).

another actual but less serious slide from my lecture:

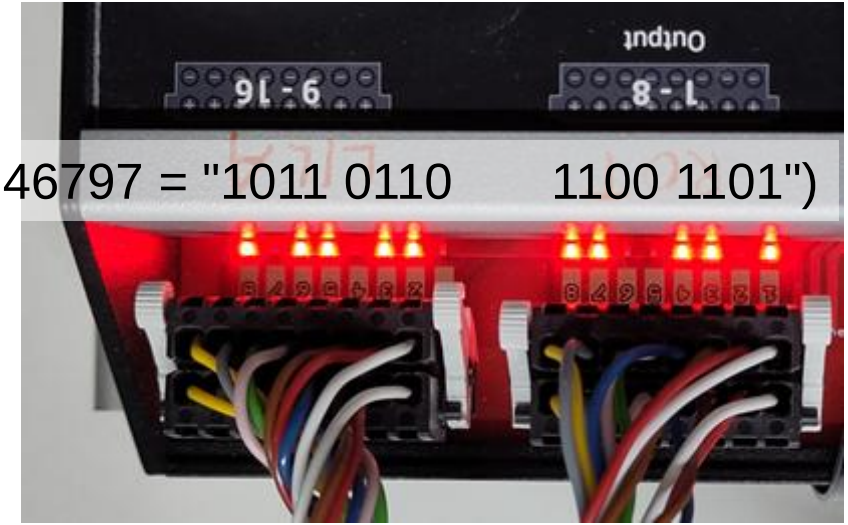


Simple way to communicate with a robot

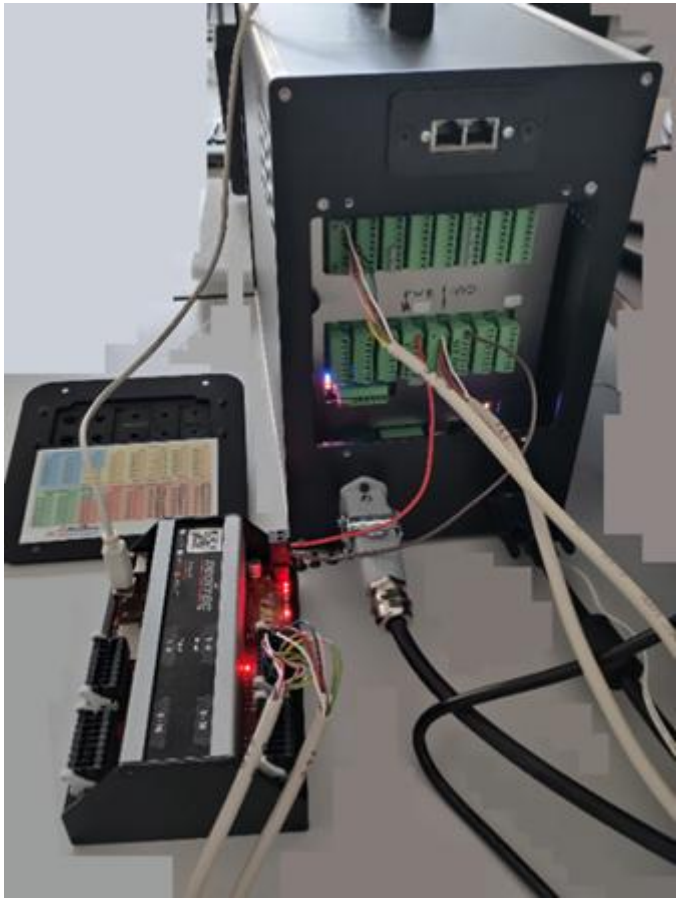
"Encoding" a signal to a robot:

the students are provided a template which sends a single 16-bit-Byte to the DIO interface (no fancy protocol for unlimited bytes)

(e.g. 46797 = "1011 0110 1100 1101")



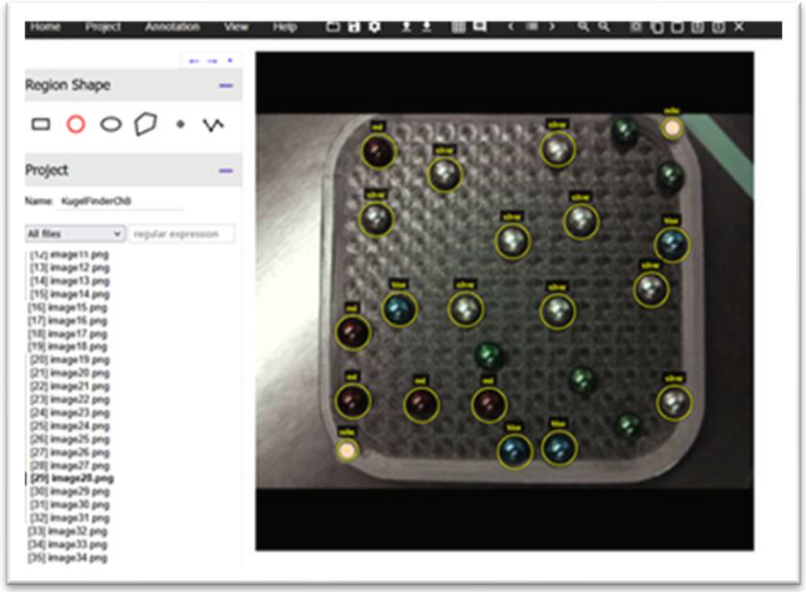
wires to the robot electronics IO panel



Challenge: The encoding on the python side must match the decoding on the robot side !

New course concept

With this prepared AI we can now expand our lecture: all other course elements remain, but YOLO and Cognex are replaced by our own AI



source: robots.ox.ac.uk

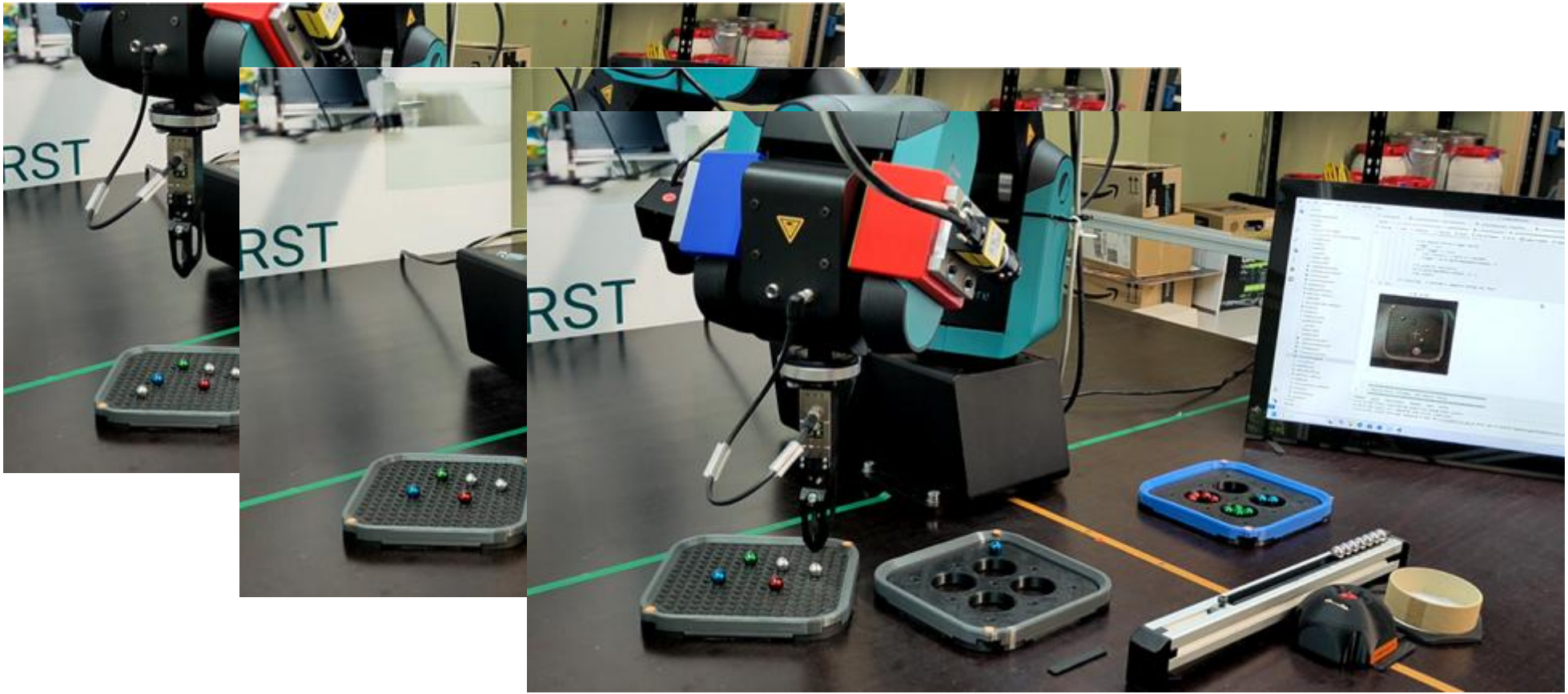
Using our own AI in our course:

1. acquire images (using a Python-code segment)
2. reshape the image with a Python-code section (at home)
3. label the images with an online-application (at home)
4. modify the python code and train the 2 AI's (at home)
5. insert the code segment to define the DIO output (at home)
6. create the RobotProgram to read the DIO input and respond properly (in the robot lab)

Estimated time to complete this assignment:
Image acquisition and labeling 2 hours; Python editing: 10 hours; programming the robot: 2 hours

The reward at the end

… and when everything works out, comes the mobile-phone moment (a little joy after hard work):



Summary

- We reported on our experiences with our previously announced method of teaching AI for machine vision and control of industrial robots
- our teaching approach to use a source-code template for the MNIST-problem proved to be highly successful and instructive within the framework of a 1 semester course
- in order to expand our course we chose not to employ open-source software
- **developed a home-grown AI module and interface to robots**
  (maybe a template for future education???)

"sour" experiences:

- the AI-software-project in total turned out to be far too complex to be developed by students at bachelor level in a single semester
- Robot-Control engineering is now occupied by 90% programming
- our IT faculty showed no interest to join in….

outlook:

We will copy this solution to multiple robots including much simpler, cheaper robots and if successful, expand this initial concept (and maybe report on this in the future….)

Thank you for your attention !