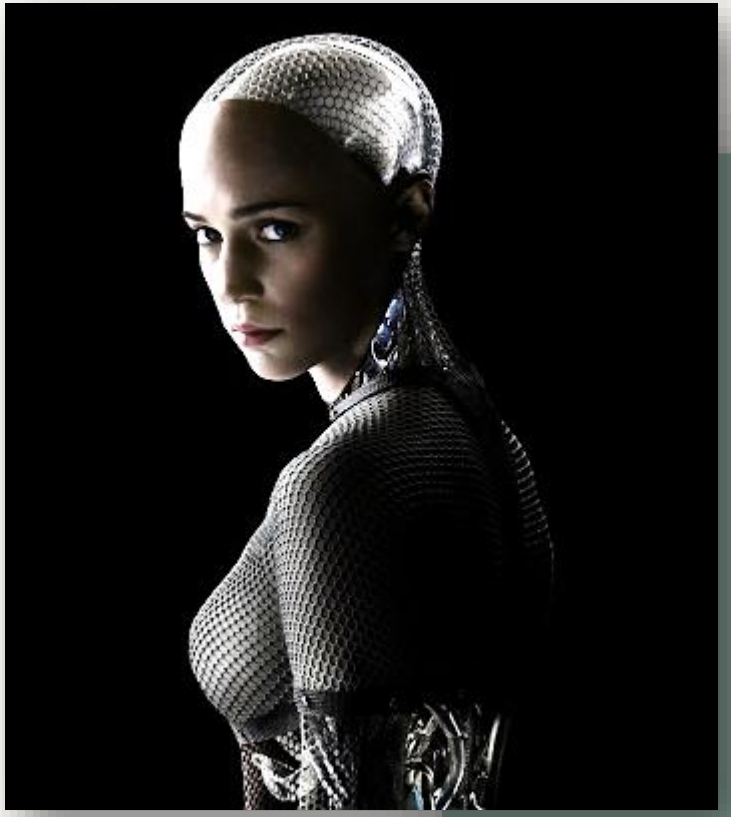
AKADEMIA
GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

INŻYNIERIA
MECHATRONICZNA

Advanced AI models in engineering

Do you even understand what you do?



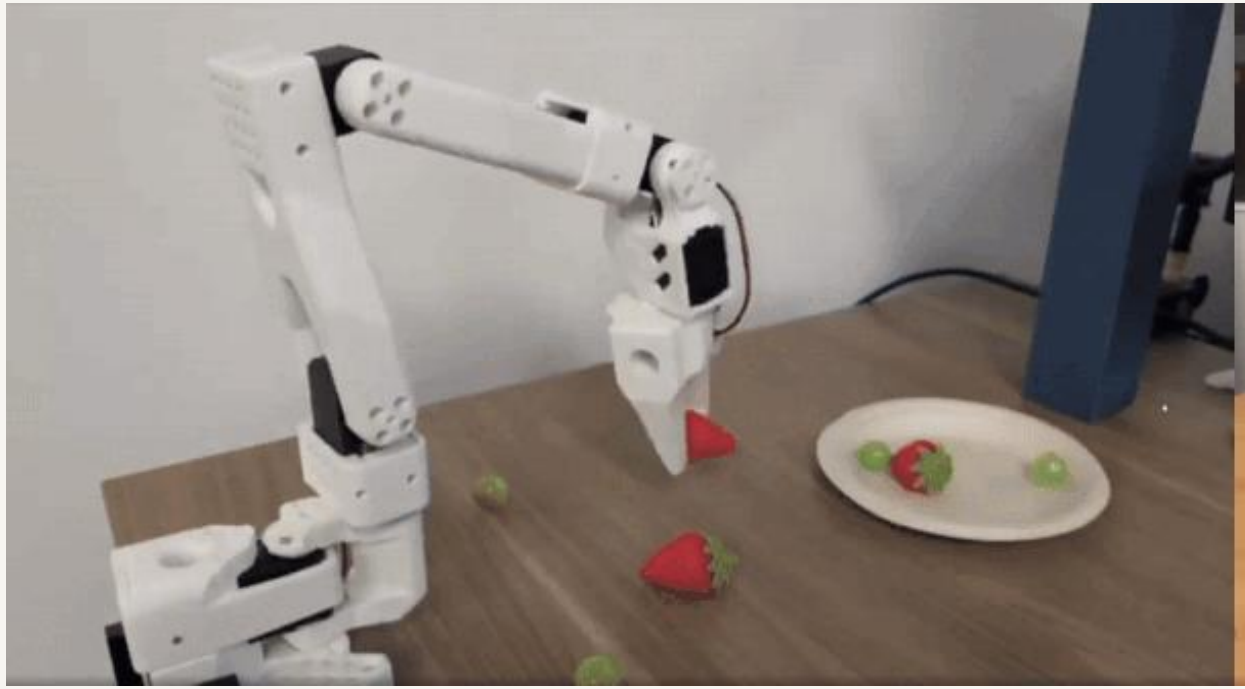
dr hab. inż. Ziemowit Dworakowski (część wykładowa)

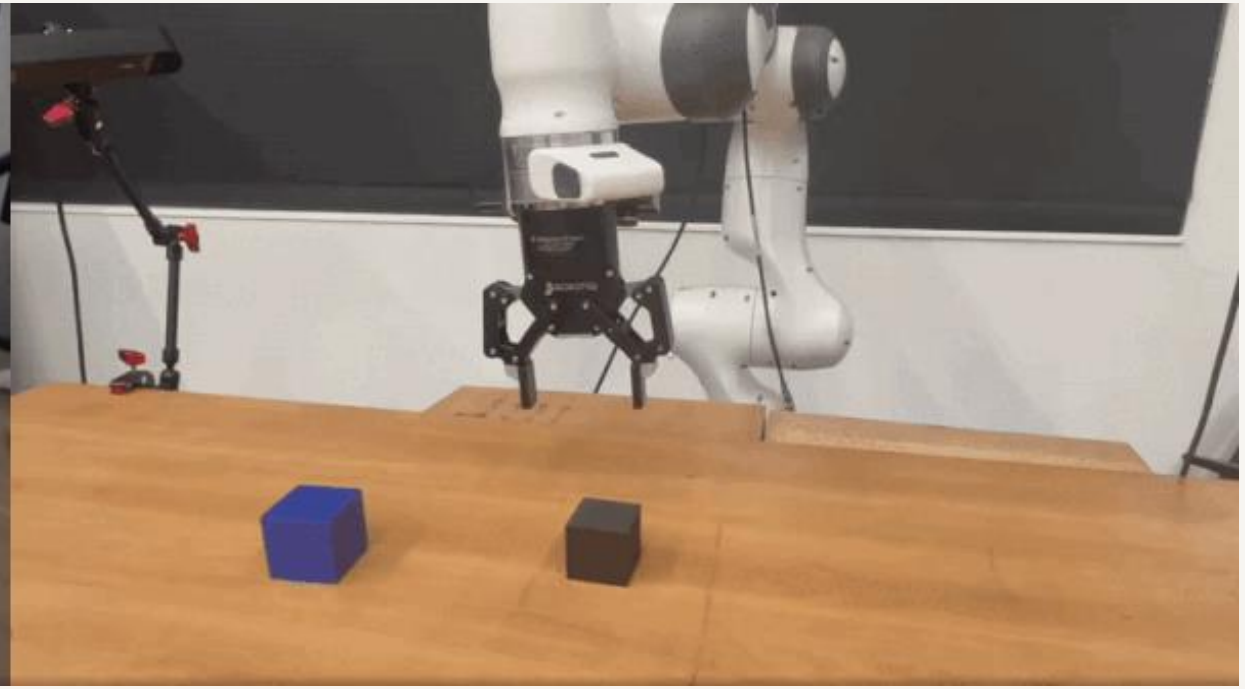
inż. Eryk Pawełek, inż. Michał Domański (część case study)

KATEDRA ROBOTYKI I MECHATRONIKI
WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI

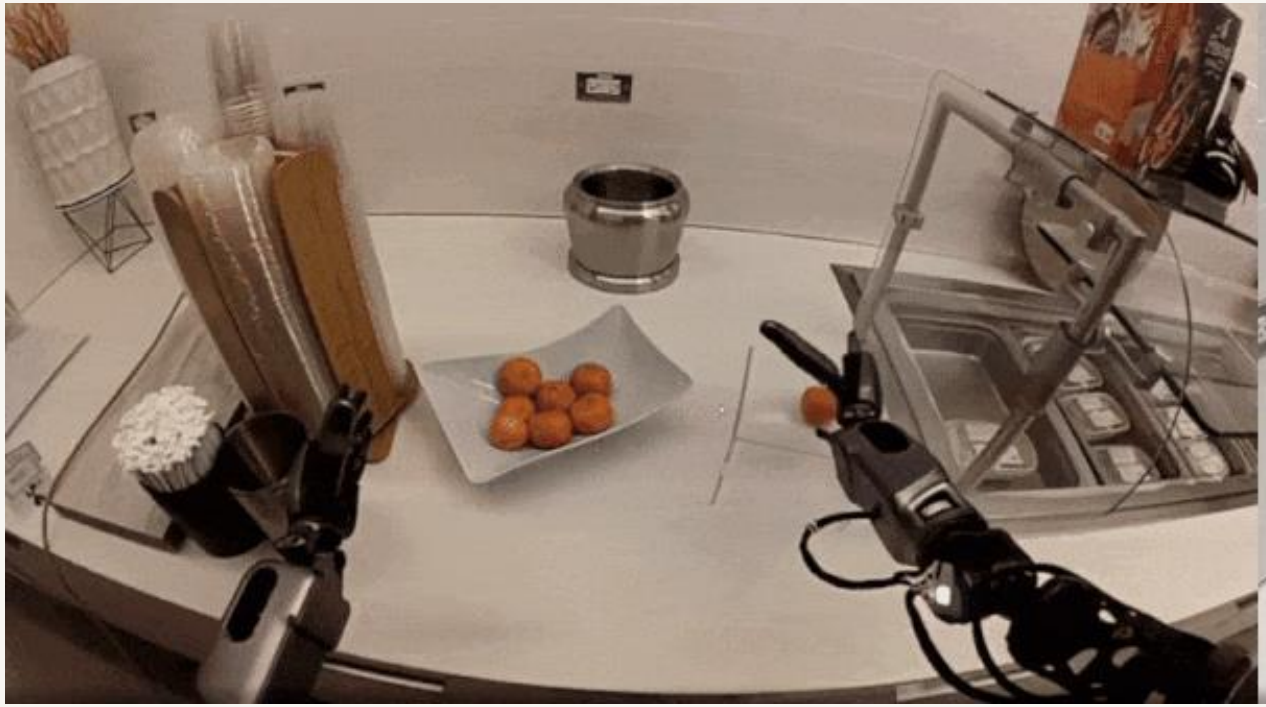
03/11/26

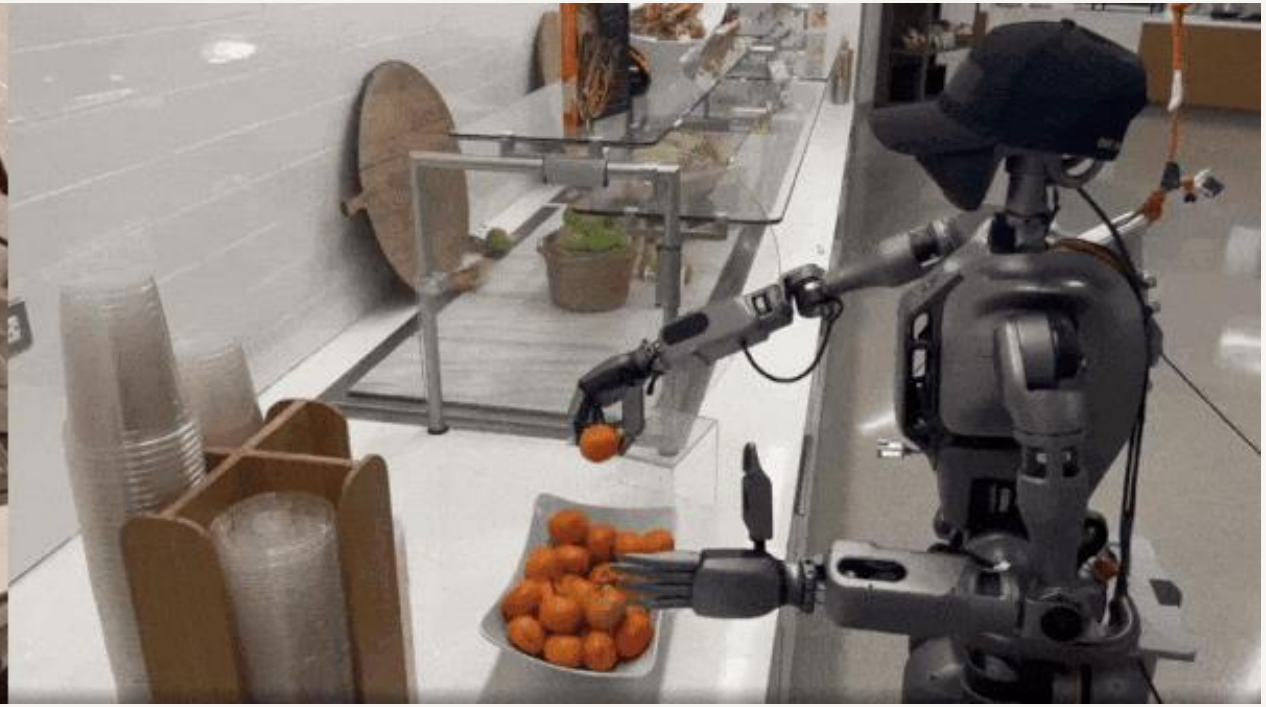
What can we use to approach this task ?



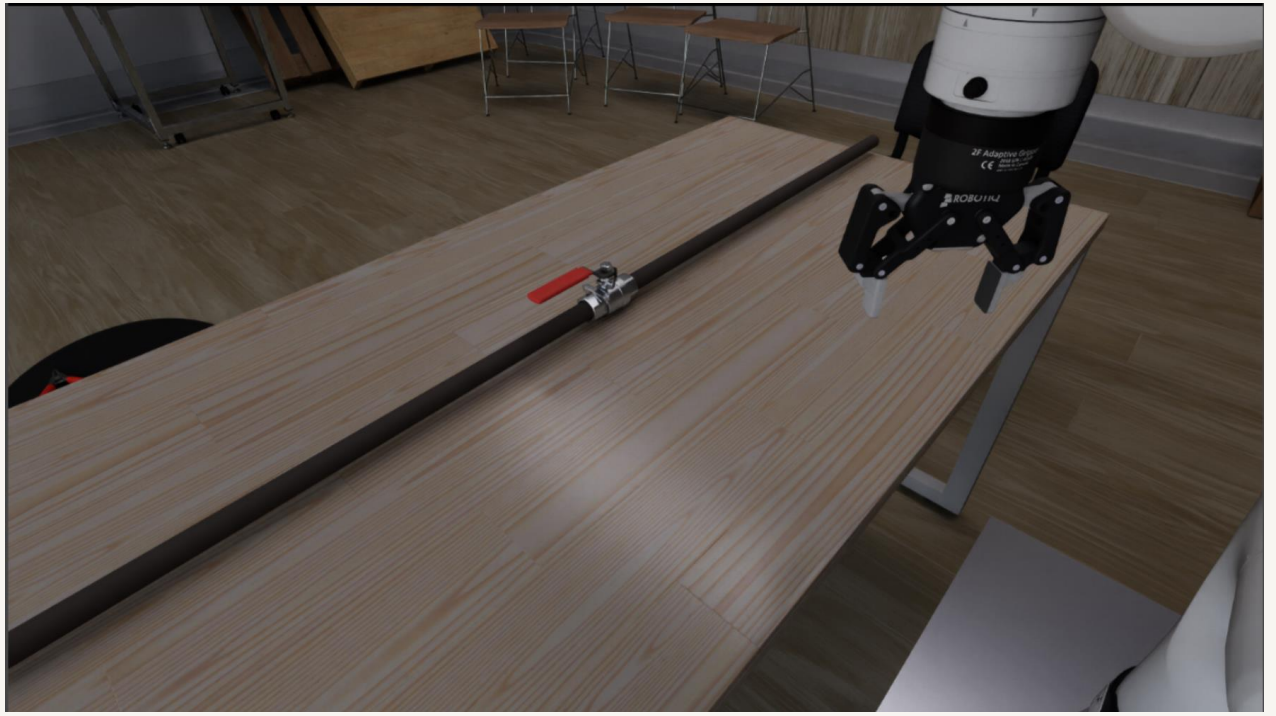


Should the solution be deterministic or not?

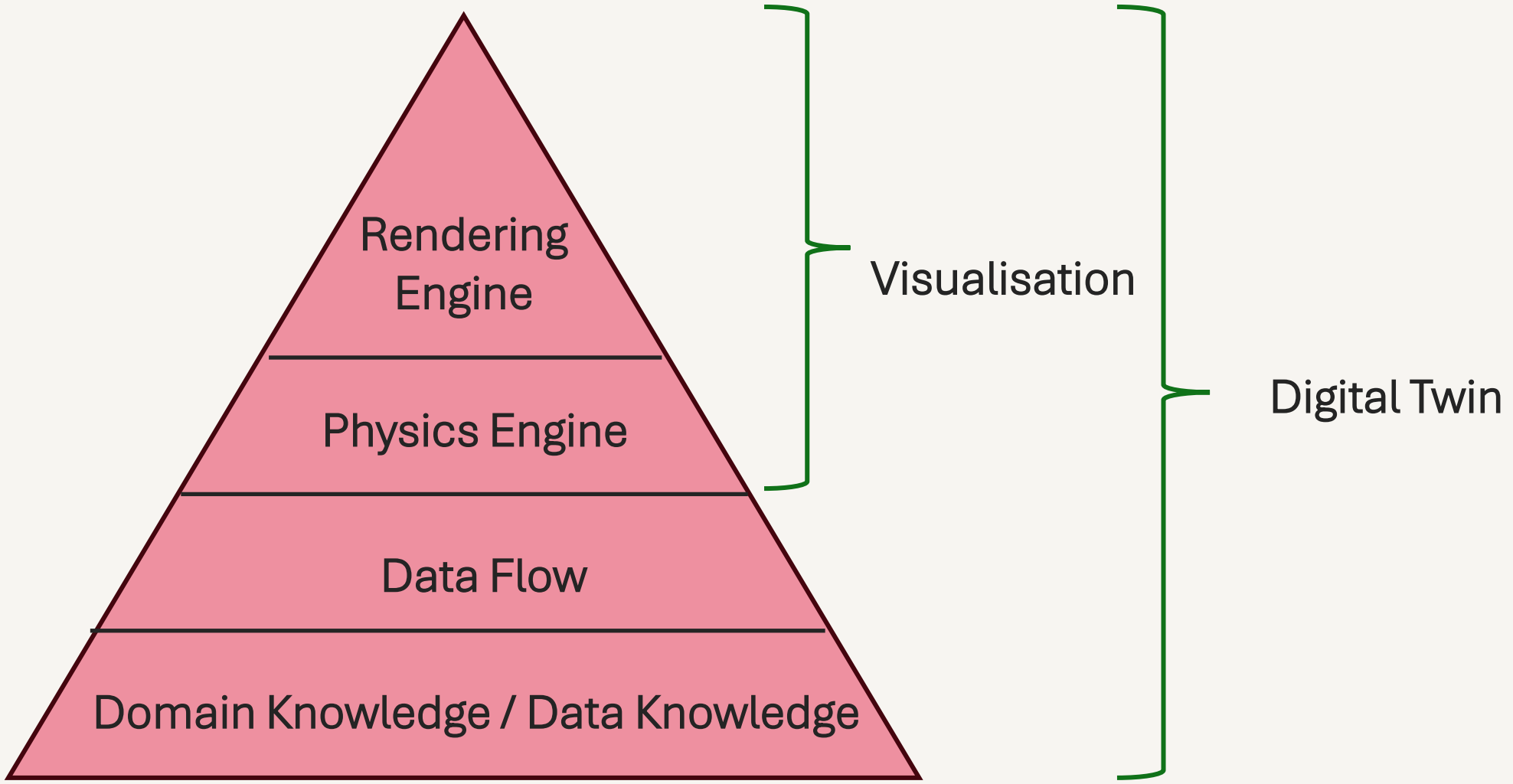


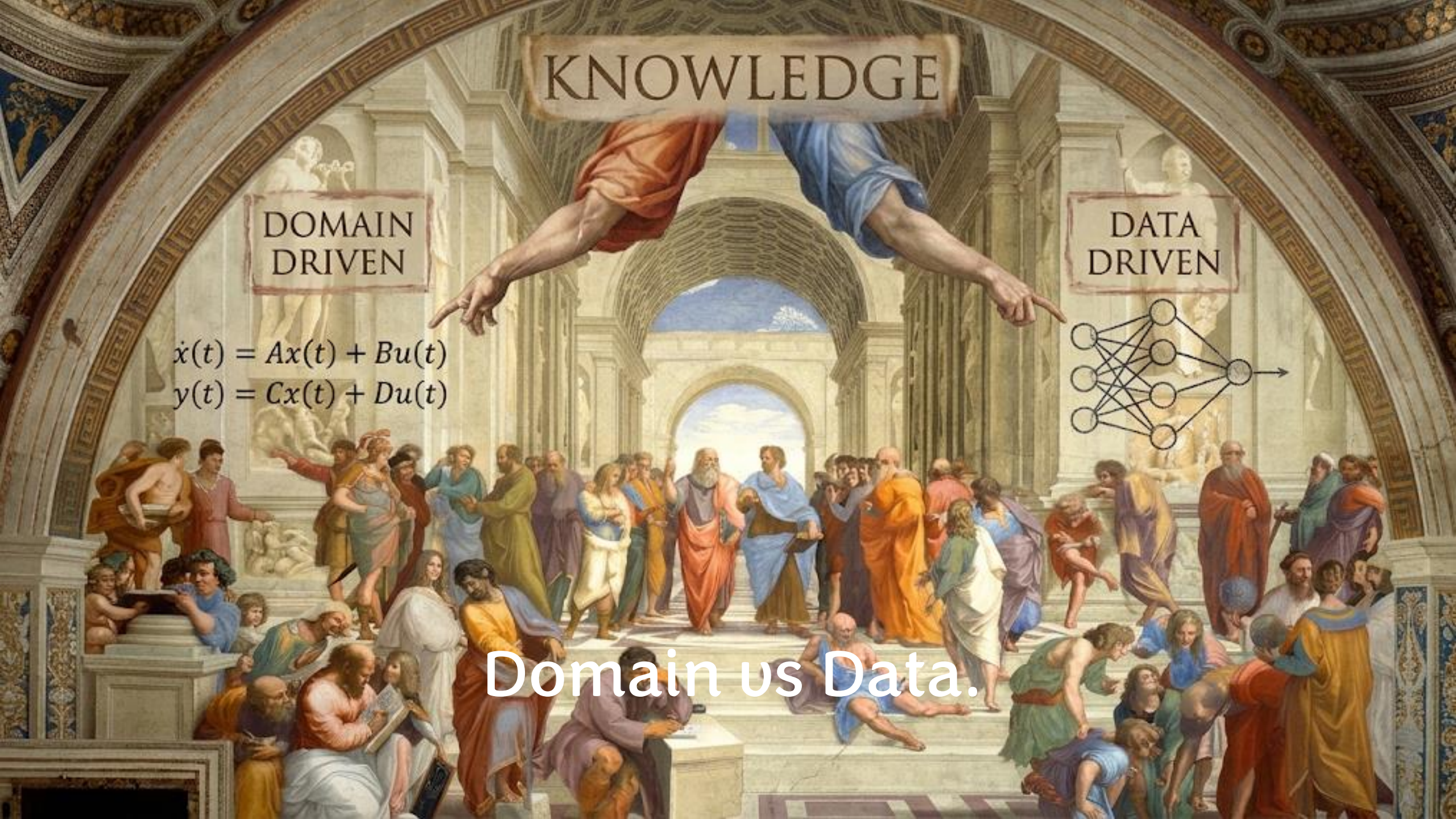




Digital Twin vs Visualization ?





Domain vs Data.

Environments



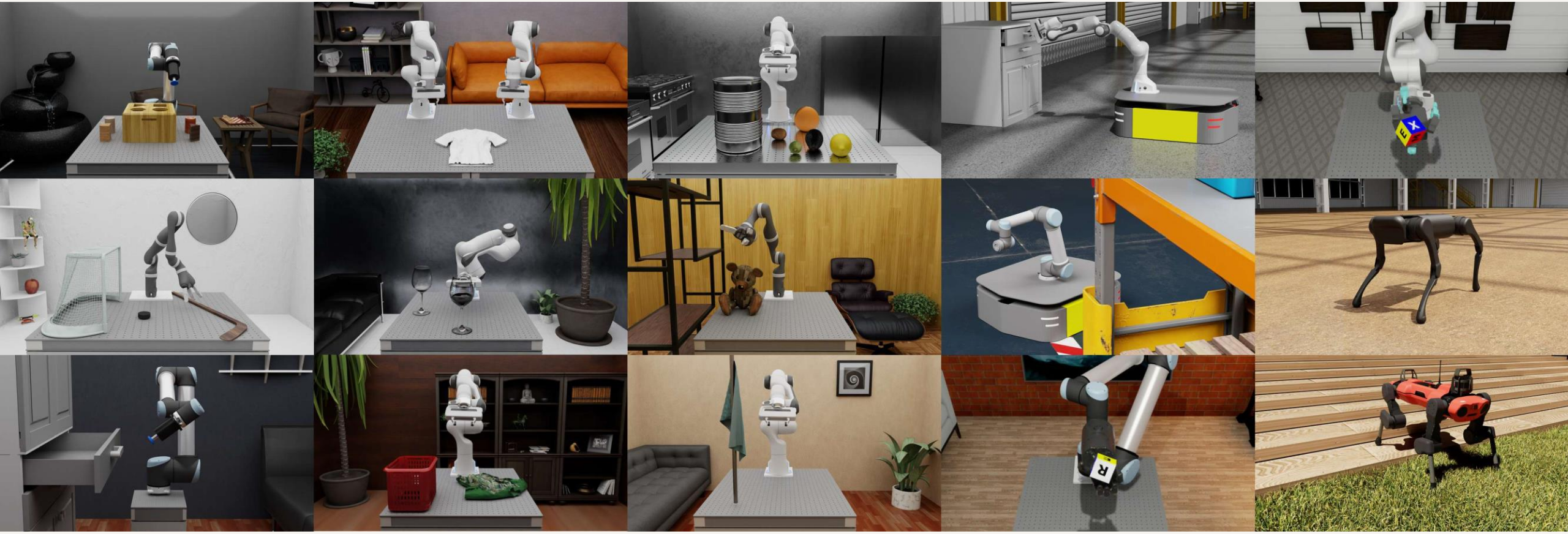








Isaac Sim.



There is a concept we use often, but have never explained during our courses...

Lots of metaparameters, potential for including context knowledge, physics-informed operators, etc... But requires having this knowledge in the first place!

„Context-related features"

„Context-related flags"

You can freely add context knowledge, physical operators and local search if necessary

In addition / **as alternative** to „getting rid of outliers" use context knowledge (and variables!) to divide data into subsets

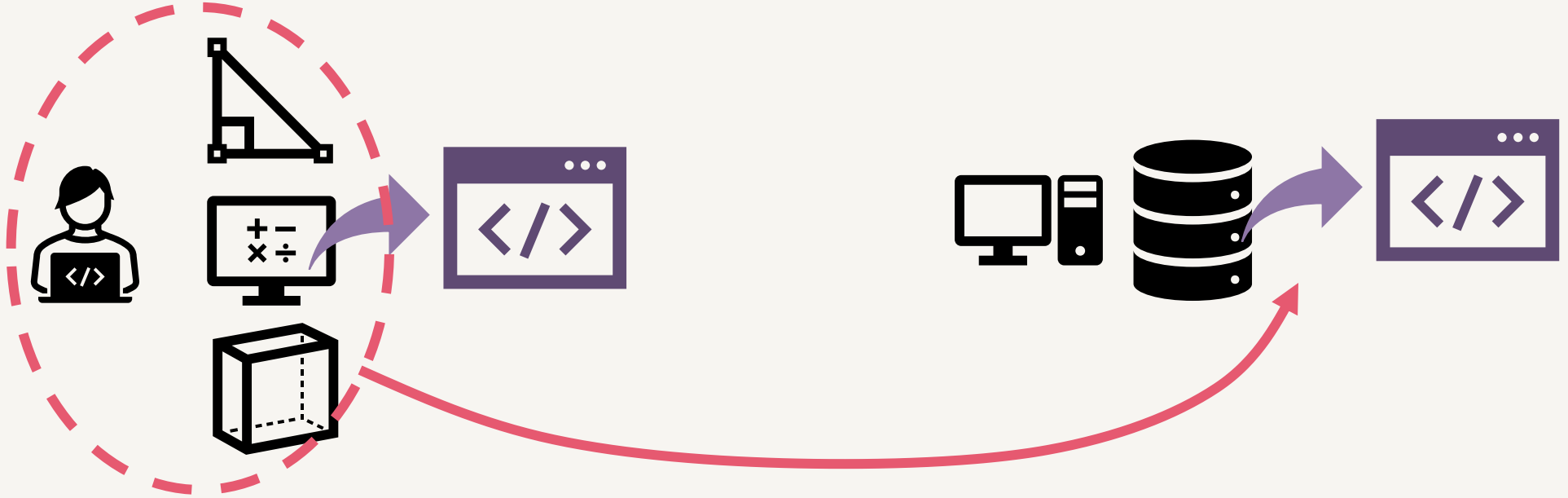
So... what is this „context" exactly?

Knowledge

Domain Driven

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

Data Driven



We use pieces of domain knowledge to make the problem easier

1st layer:

Physically possible numbers

(features, outputs,predictions)

FIX

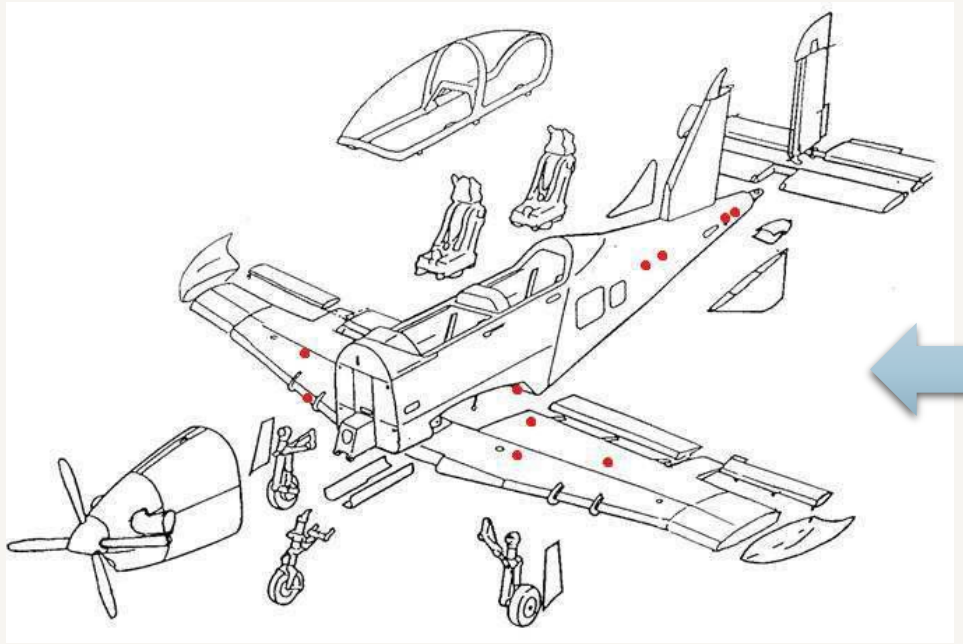
Concept 2: „sanity check”

If context tells you what to expect (e.g. in what ranges the measurements are supposed to be) use that at first layer of decision – to check if measurements are valid

„power generated can’t be negative”

„elevator speedshould not exceed 999 kph”

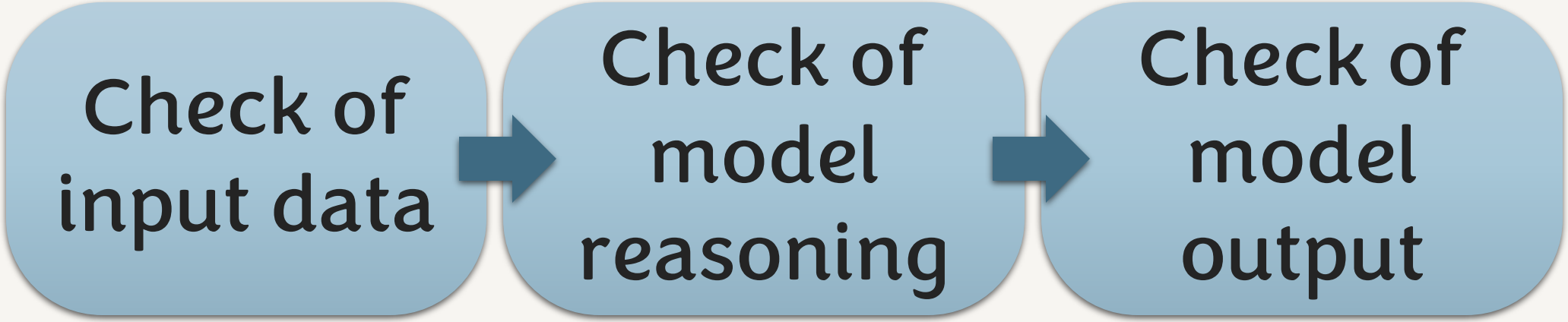
*„mean of **vibration signal** should be close to 0”*



That’s what we did here!

Its unlikely that ALL the measurements suddenly went off!

Sanity checks – at all levels of model use



„Are these measurements possible?”

„Can a decision space look like that?”

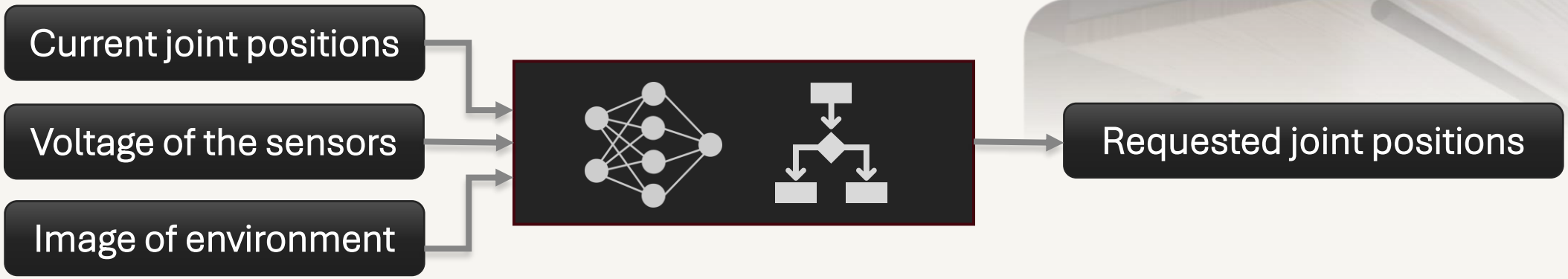
„Is this answer physically plausible?”

Validation of input measurements
(against e.g. faulty sensors)

Constrained decision space
(we know where the table is. We can't move through it)

Checks of the outputs
(can the joint move with this velocity?)

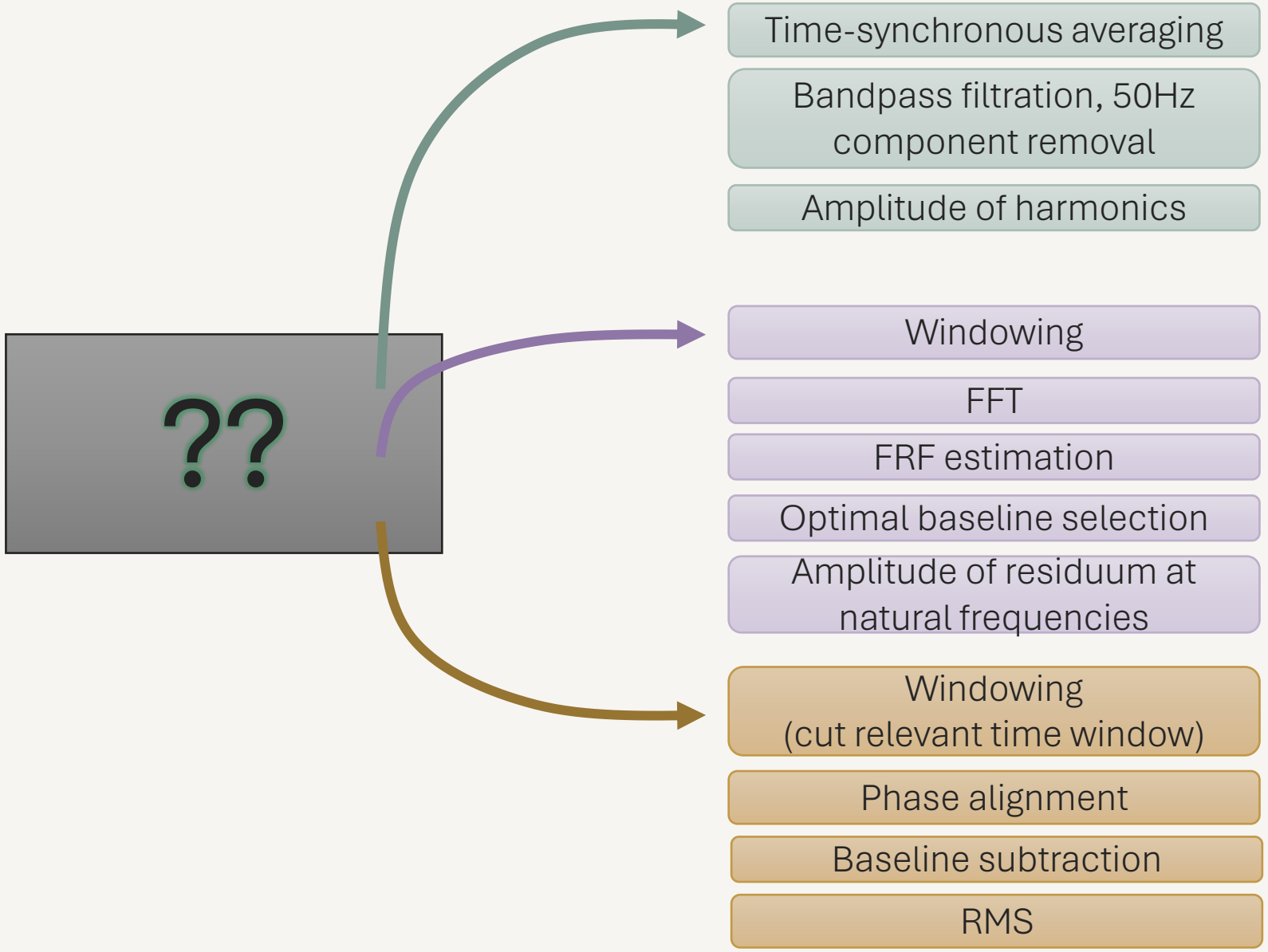




Benefit: *the mapping can be imperfect and robot will still „not fail completely"*

2nd layer:
Physically motivated features
(based on understanding of physics)

Context-informed parametrization



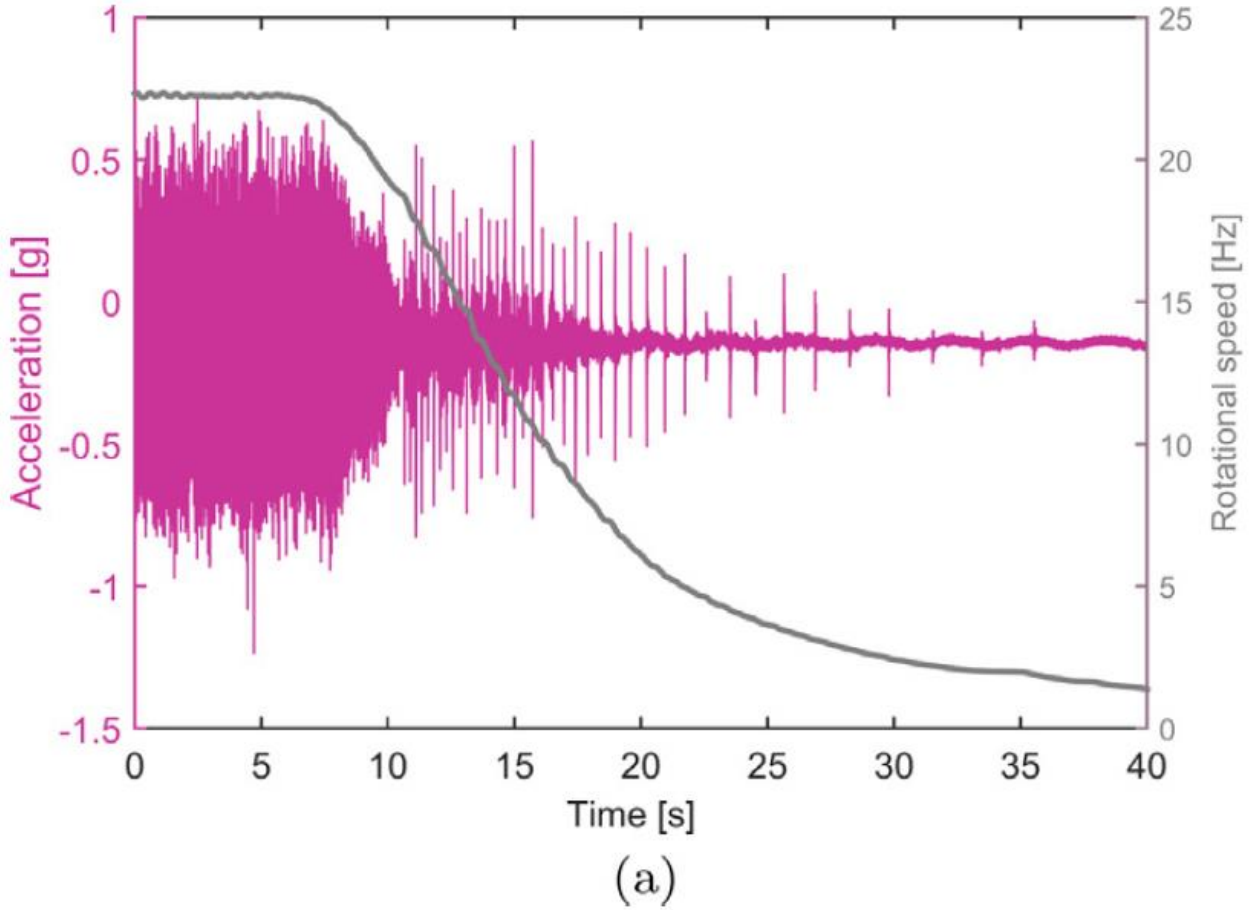
Condition monitoring, gears, steady speed

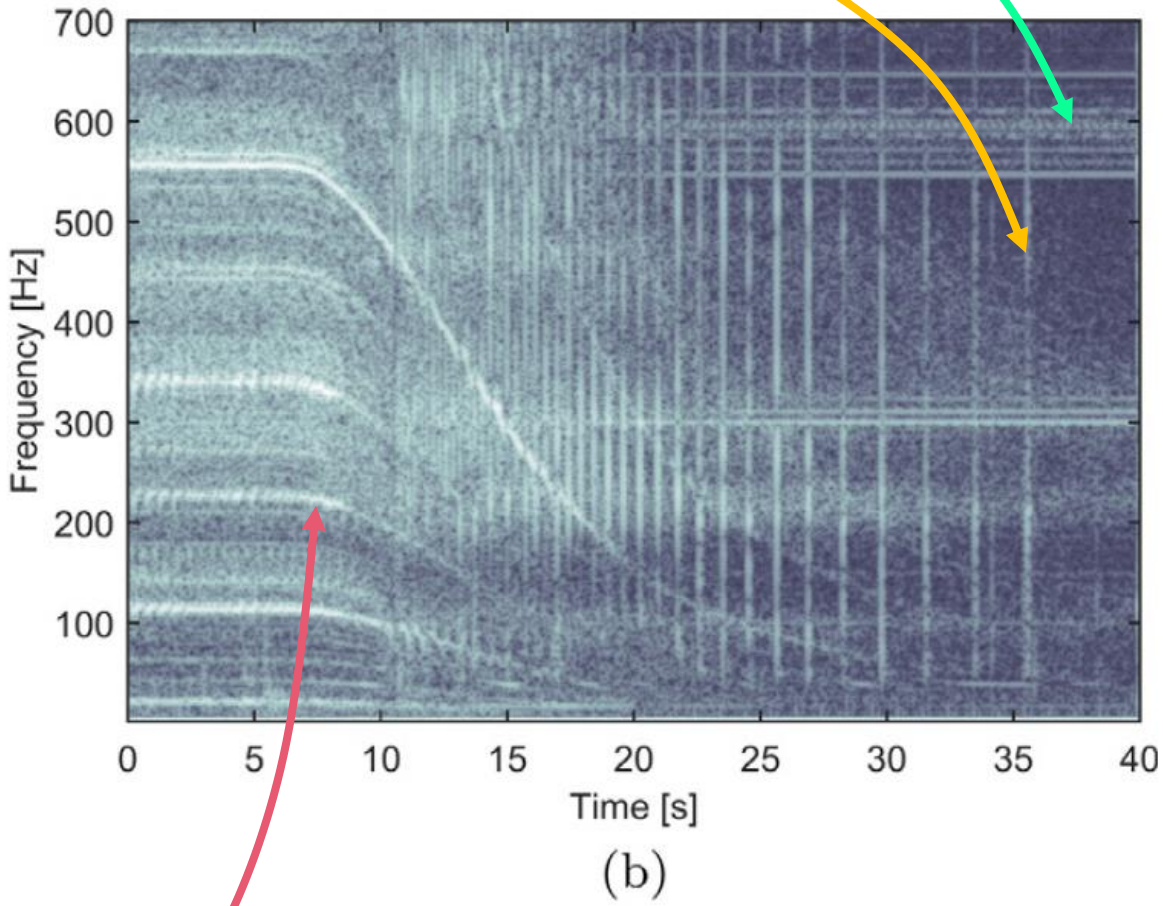
Long-term structural monitoring through modal analysis

Guided waves in variable temperature

Unrelated with machine's operation...

Impacts (once per revolution)





Characteristic frequencies + harmonics

You already introduced context knowledge almost every time when doing parametrization!

You can do that on several levels:

Note, that this is a choice that is irreversible!
Any model placed downstream from this decision can't recover these...

Deep-learning-usable

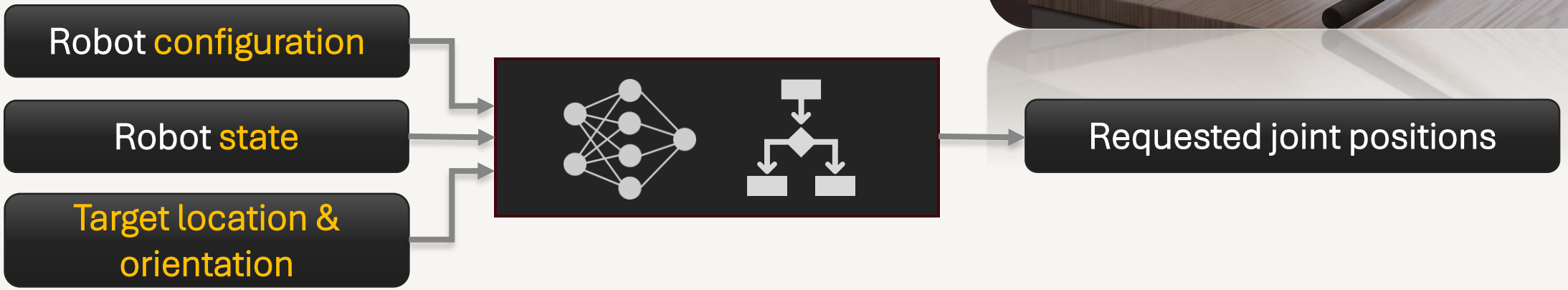
- Preprocessing of data to highlight known features or supress unwanted information
  - *e.g. filtration of 50 Hz component, phase alignment, time-synchronous averaging*
- Domain changes to make follow-up feature calculation easier
  - *e.g. fast fourier transform, order spectrum, modal coordinates*
- Choice of features to calculate (and not to calculate)
  - *e.g. RMS, correlation, amplitude of peaks*

Physics-informed features
(e.g. cartesian coordinates)

Target location through image features
(we know what the target is, lets calculate its position)

Features calculated from sensor voltage
(e.g. is the effector closed or opened)





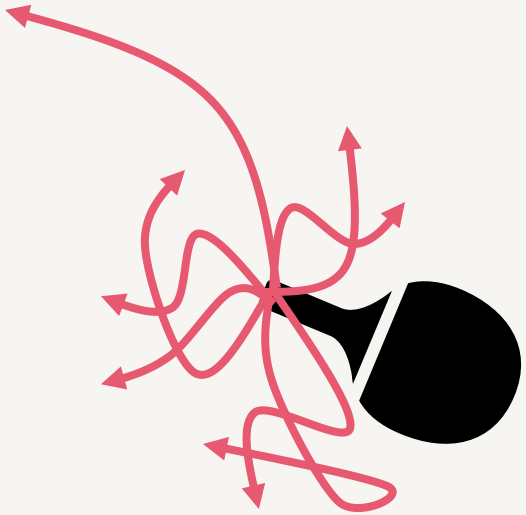
***Benefit:** We need much, much less data to reach convergence.*

But we require more constrained environment

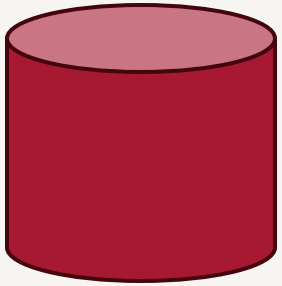
3rd layer:
Physically informed training goals
(grounded in our process understanding)

Layered loss functions
(„reduction of the scaffolding")

Thousands of tries – and we may not even hit the ball once!







ANN gets a point when ball stays in a basket

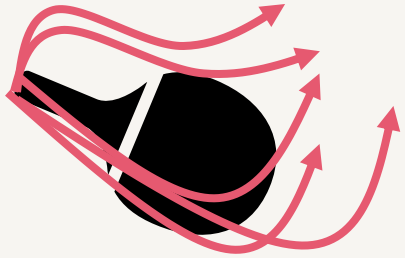
Layered loss functions
(„reduction of the scaffolding")

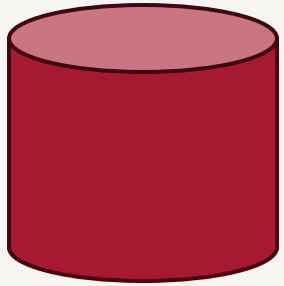
Let our loss (goal for minimization) be:

$$L = w_{rp} \cdot P_{rp} + w_{bb} \cdot P_{bb} + w_t \cdot \frac{1}{T_b}$$



First: minimize proximity of racket to ball P_{rp}





Second: minimize proximity of ball to basket P_{bb}

Third: maximize time spent in basket T_b

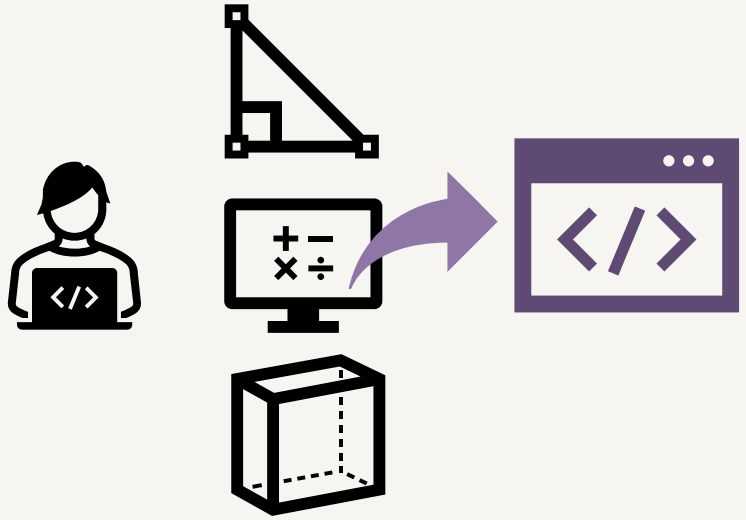
Usage of expert knowledge

„M takes always twice as long as N”

„When X happens, we need to do Y”

„Don’t exceed Z value”

„We never allow for X to happen!”



We can:

Use fuzzy logic system with expert rules implemented straighforwardly

Use expert to adjust loss function weights

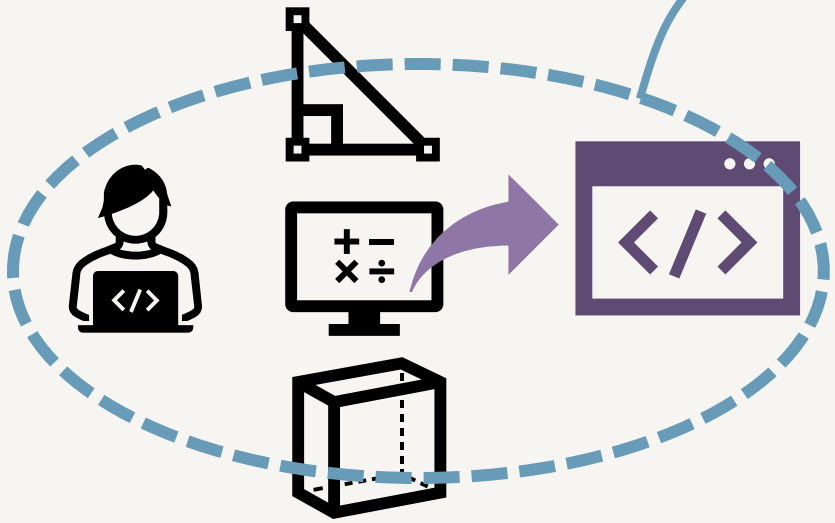
Use expert to rate our attempts (as a „final loss function”)

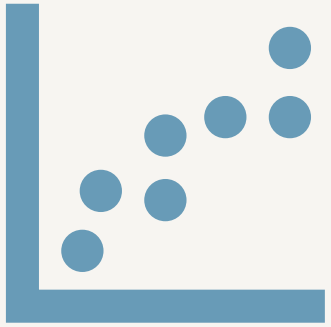
Watch for lies and misconceptions though!

Automatic transfer learning from an expert

We recreate
(what outputs should be obtained for what inputs)

„expert" does not need to be human!





We observe
We store input-output relations

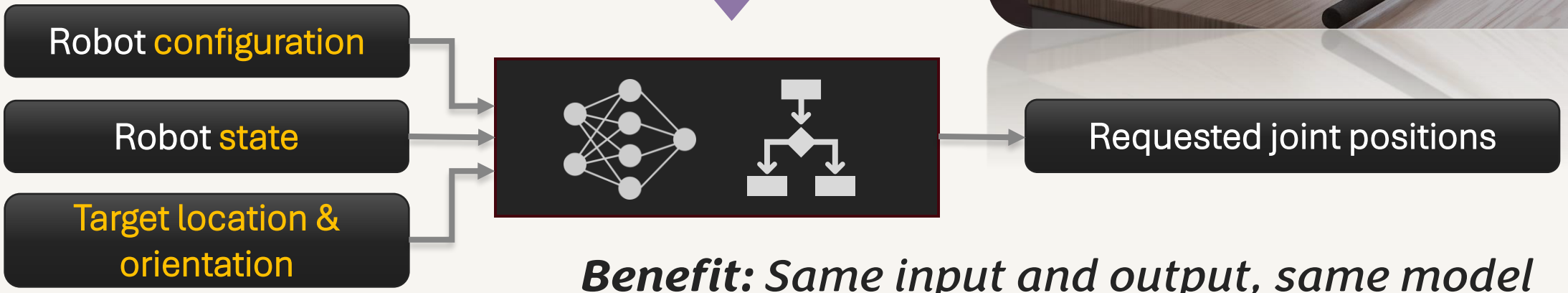
We can be only as good as an expert is
*- **so this mapping serves as a starting point for further improvement***

Layered loss function

(e.g. we first train the robot to poke the target. Then we train it to grab it)

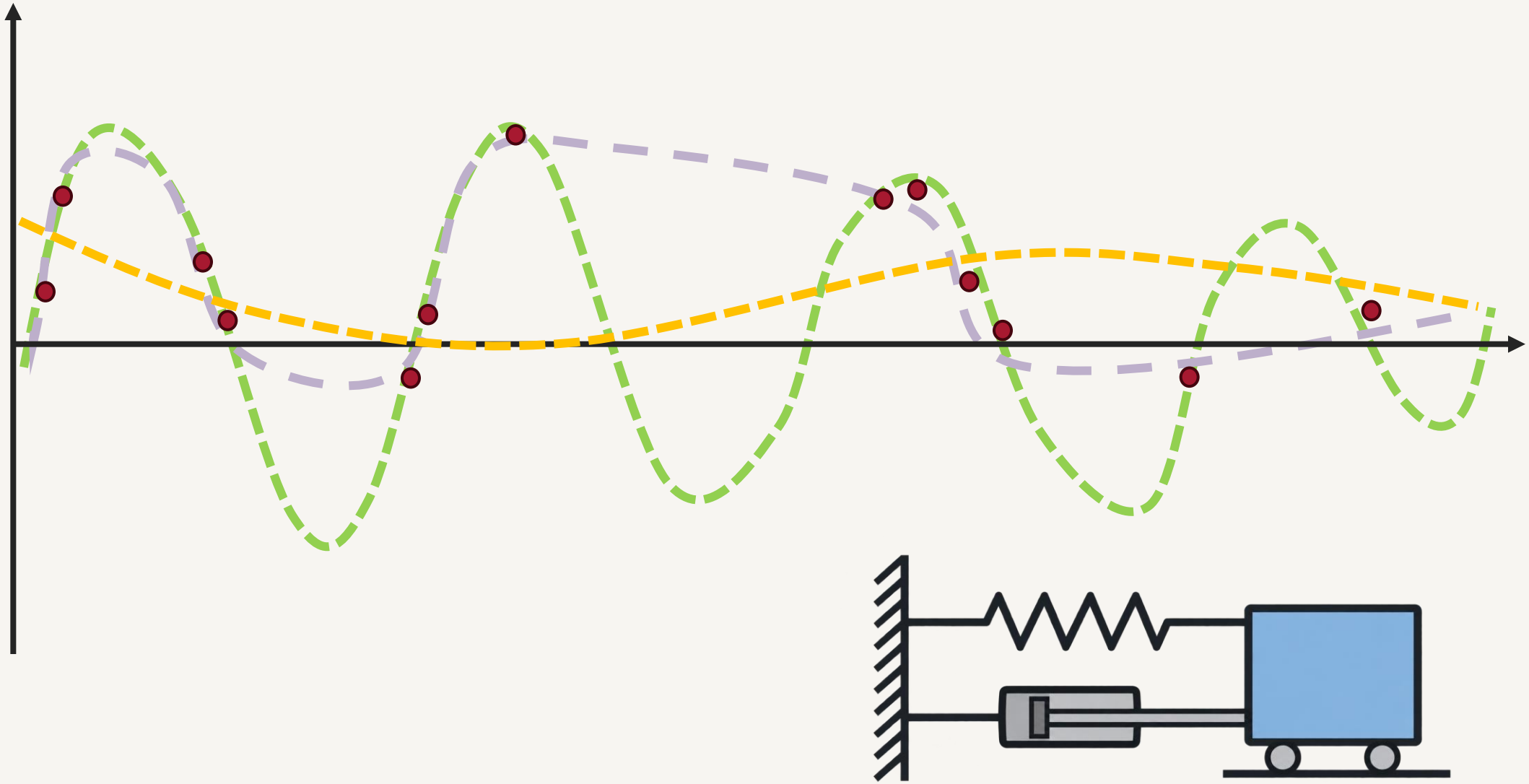
We gather 100 input-output sequence mappings by human operators

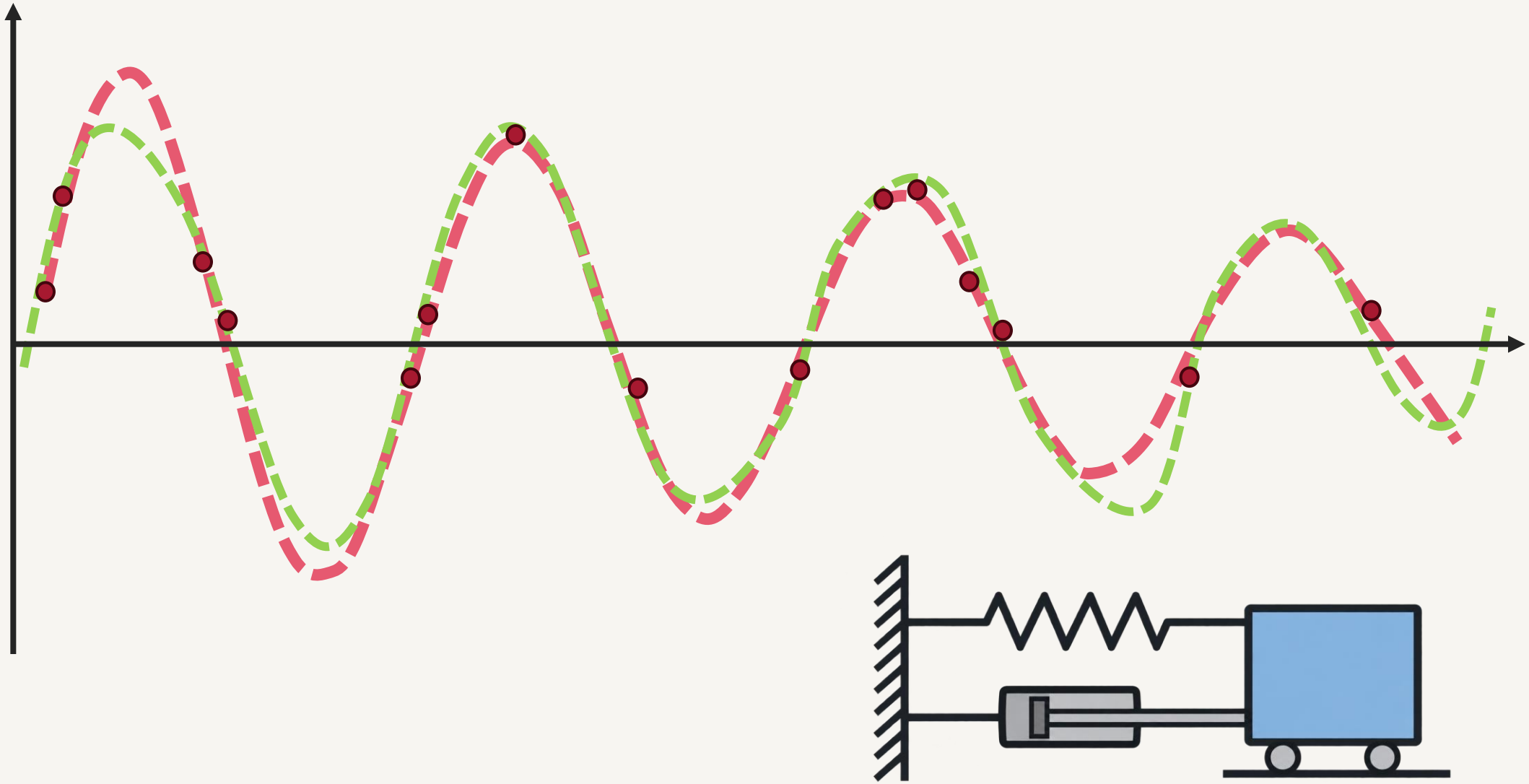




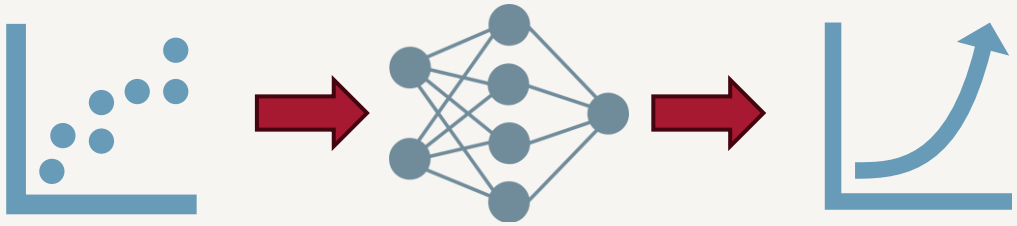
Benefit: *Same input and output, same model structure – but much better starting point for training*

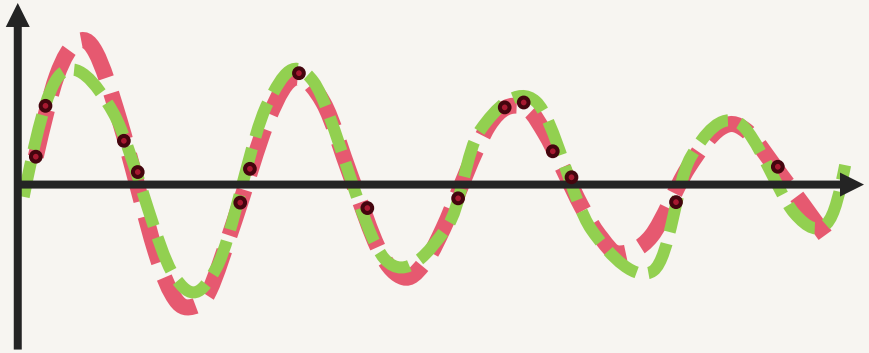
4th layer:
Physically informed reasoning
(based on understanding of physics)





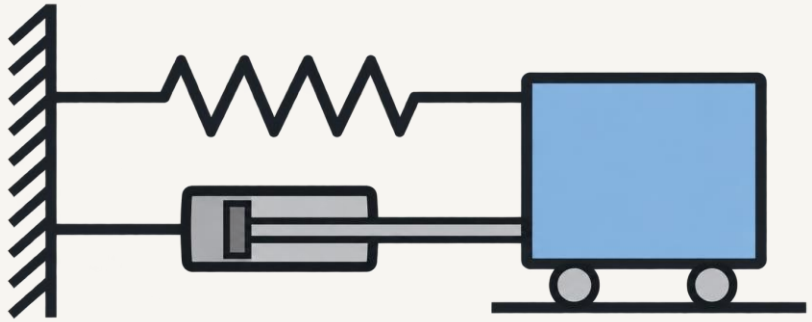
Physics-informed Neural Networks (PINNs)





1) Model proposes a solution *(here: trajectory)*

2) We run the solution „through physics”
(here: calculating velocity and acceleration)

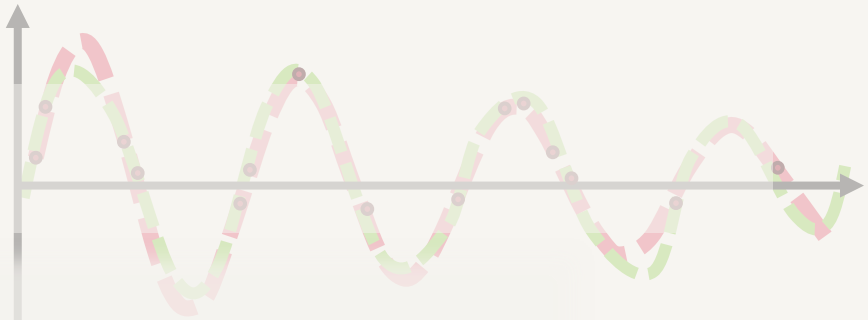


3) We calculate residuum (what part of the solution is not physically plausible)

4) We use this residuum as another criterion for the model training

Physics-informed Machine Learning





We can also:

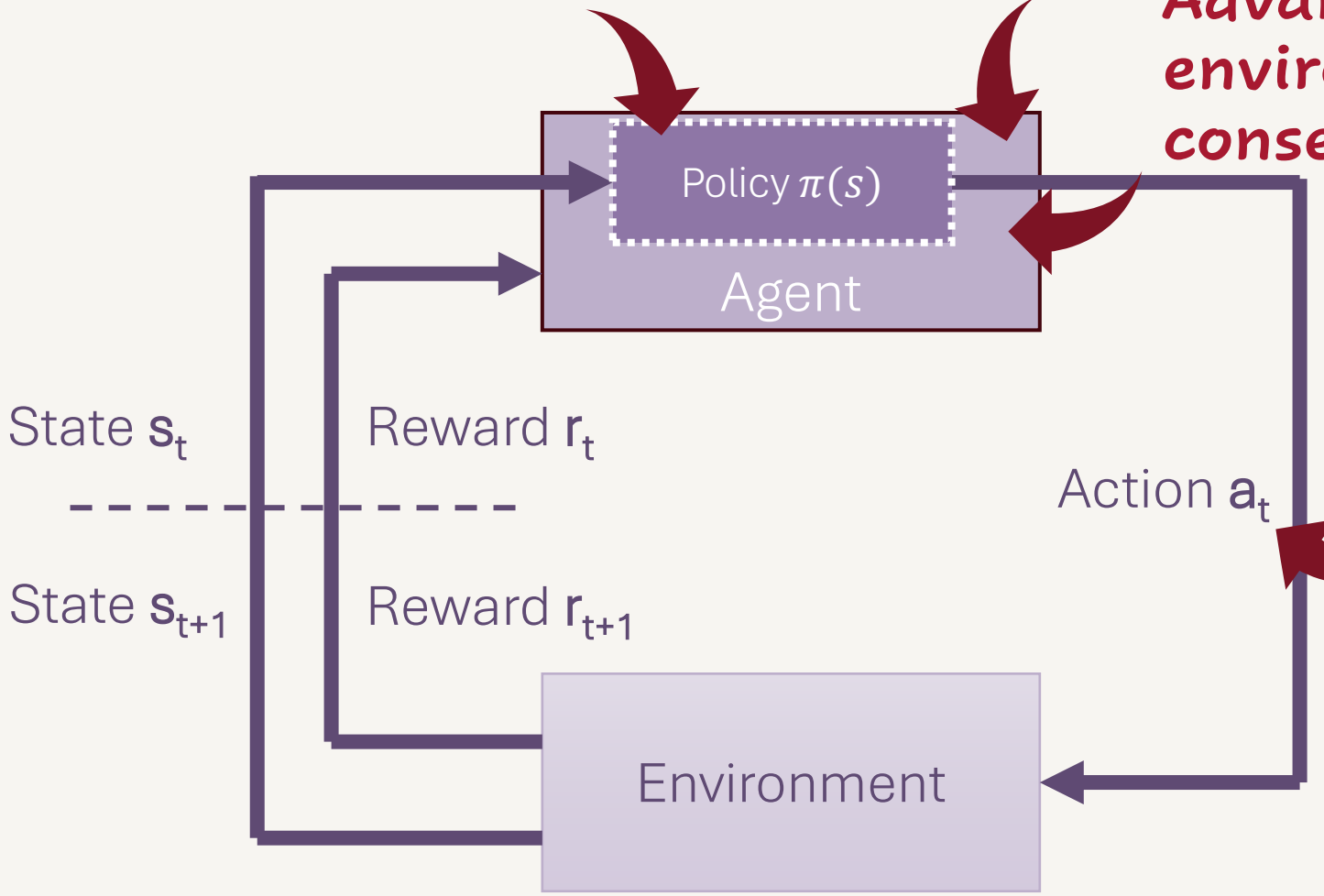
- **Force network structure to follow differential equations' structure (e.g. nodes representing system states)**
- **Use activation functions in some layers to mimic behavior of particular physical systems**
- **Make the network predict system states and calculate the rest from the physical model (e.g. here model finds m, c, k and x_0 parameters, rest is calculated from the equation of motion)**



Main decision mechanism

Agent updates the policy

Advanced agent can model its environment predicting consequences (and rewards)

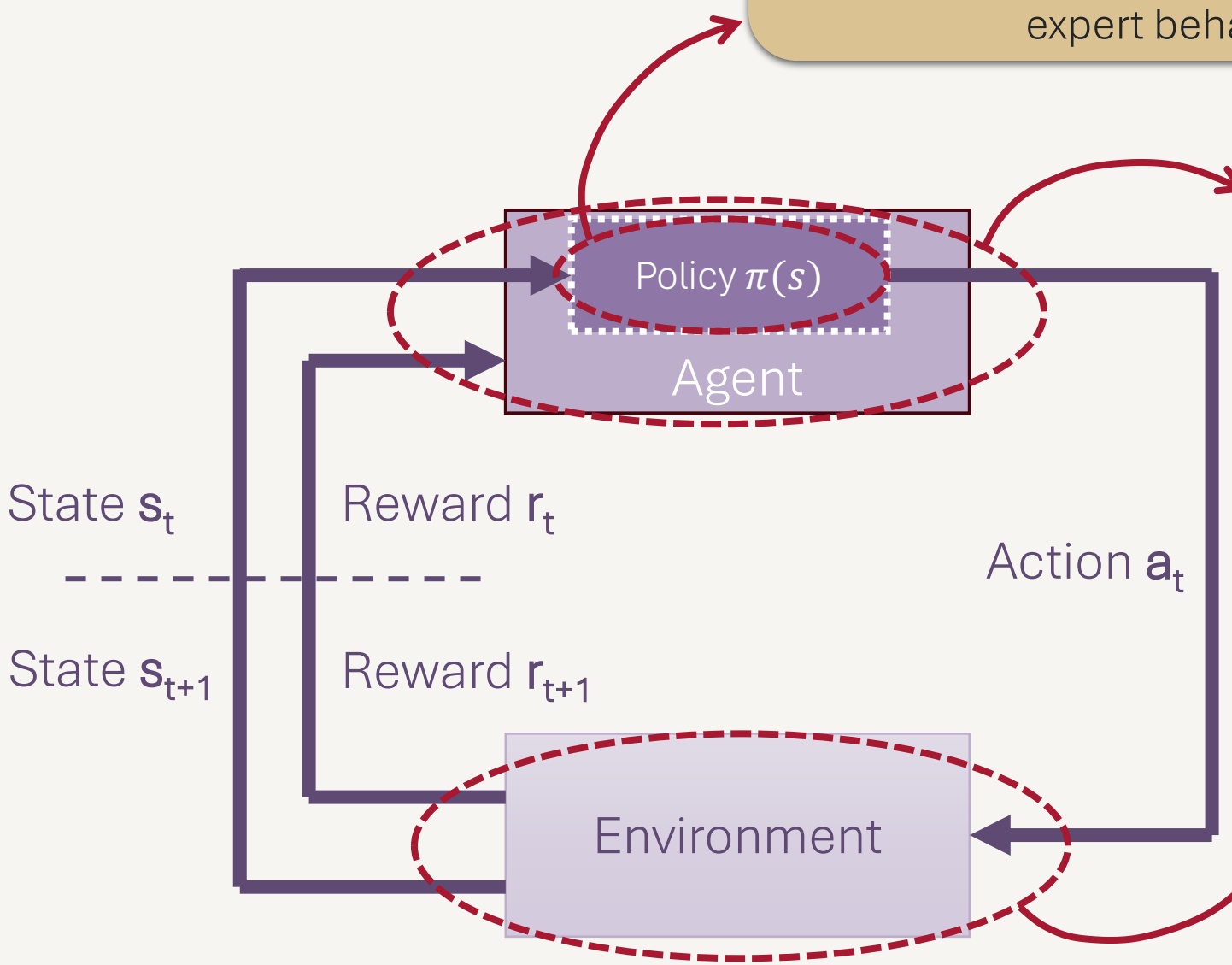


If action precedes reward, its gets reinforced

Policy can be physically-informed
(no impossible moves)

Initial policy can be trained to mimic
expert behavior

Agent can model environment through
context-informed models (e.g. PINNs)



Here we can have either:
- **physical measurements** (costly!)
- **numerical simulations** (tedious setup)
- **surrogate ML model fitted to reality**

Model structure allowing for physics-informed relations

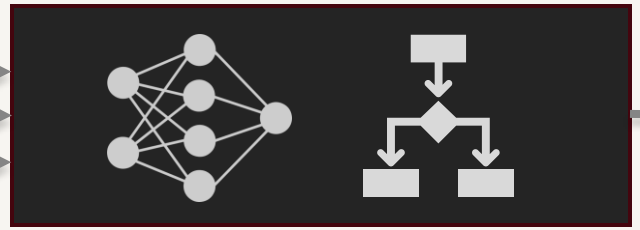
Digital twin
used for training data for the system



Robot configuration

Robot state

Input image



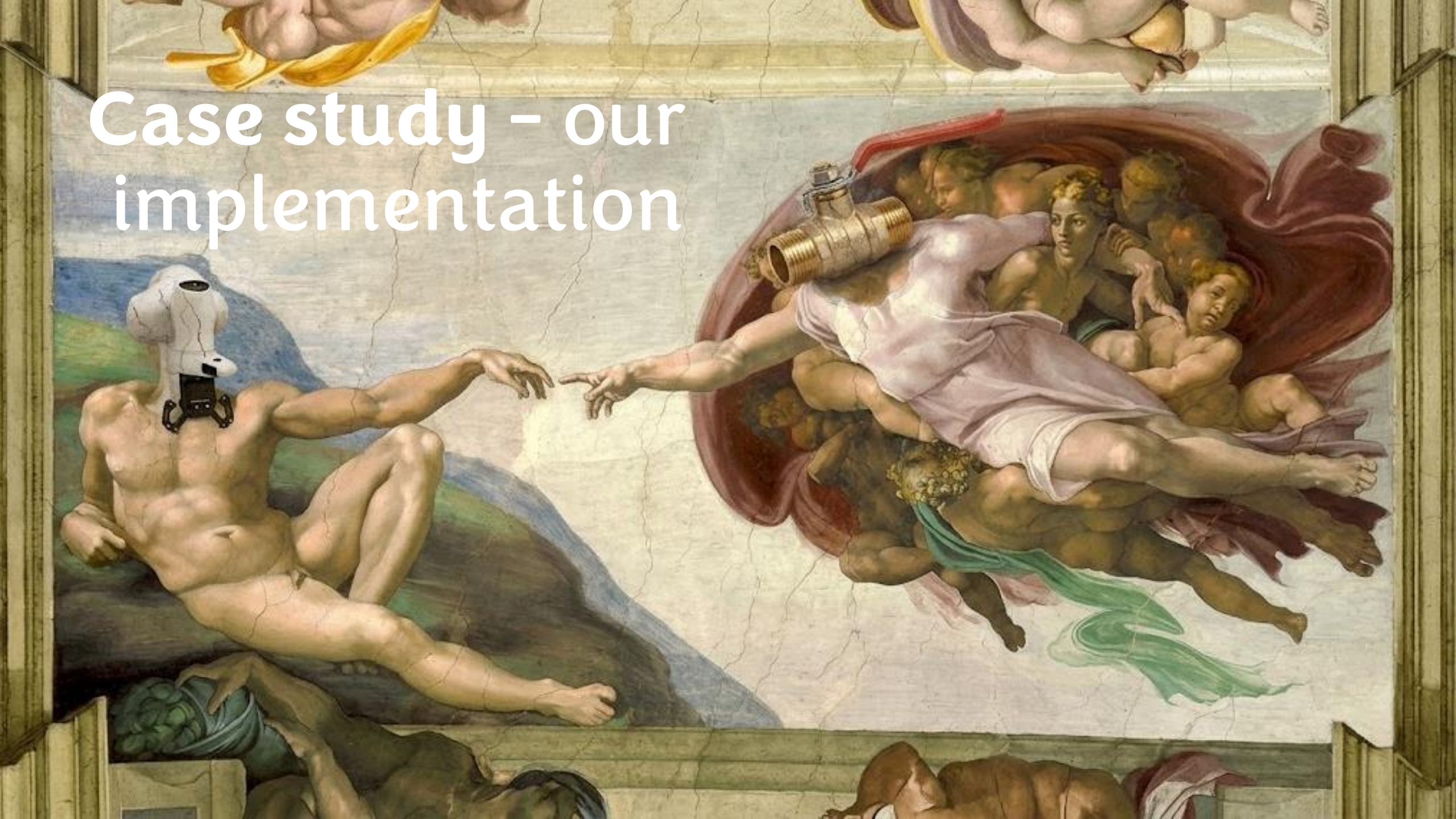
Requested joint positions

And now we can reduce constraints

Benefit: *Model generates physically plausible outputs not because it was trained this way, but because there is no other possibility*

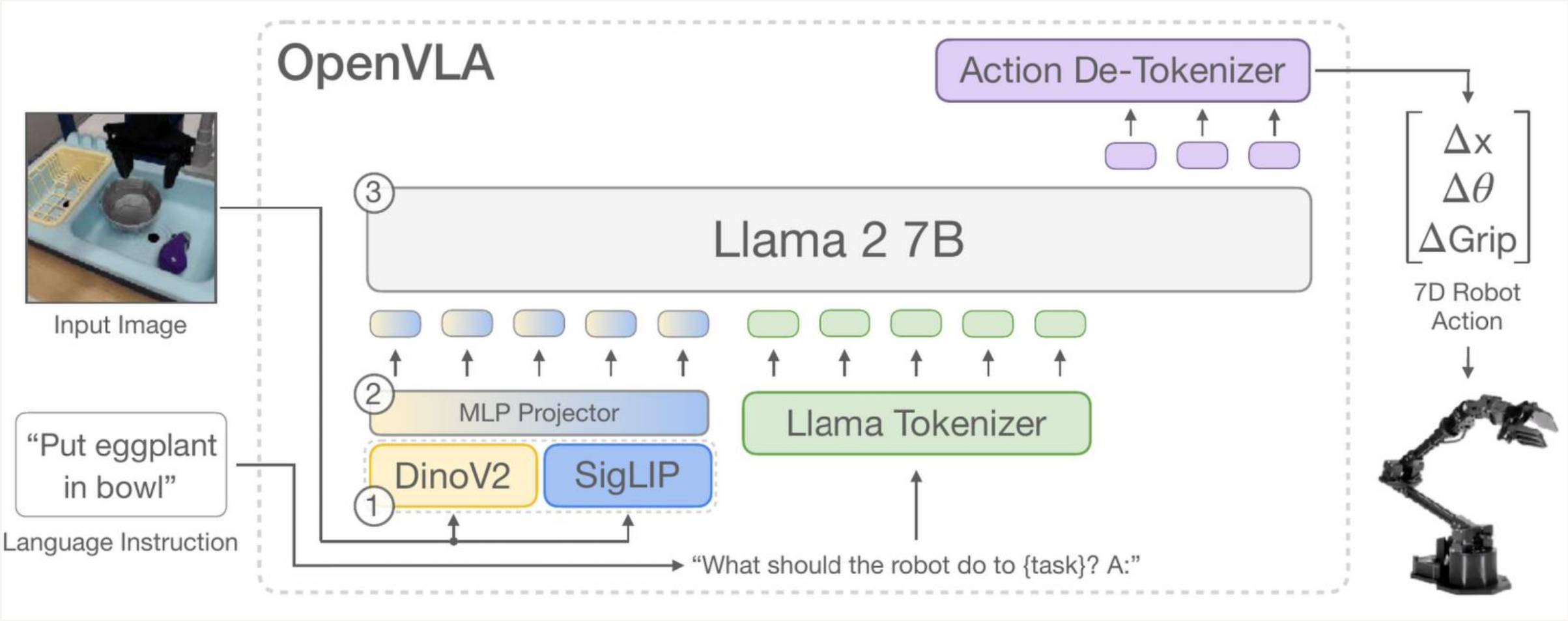
What’s the actual setup we are working with?

Case study – our implementation



General Purpose Robotic Models

VLA vs WAM



Source: M. J. Kim, K. Pertsch et al., "OpenVLA: An Open-Source Vision-Language-Action Model", 2024.

Implementation





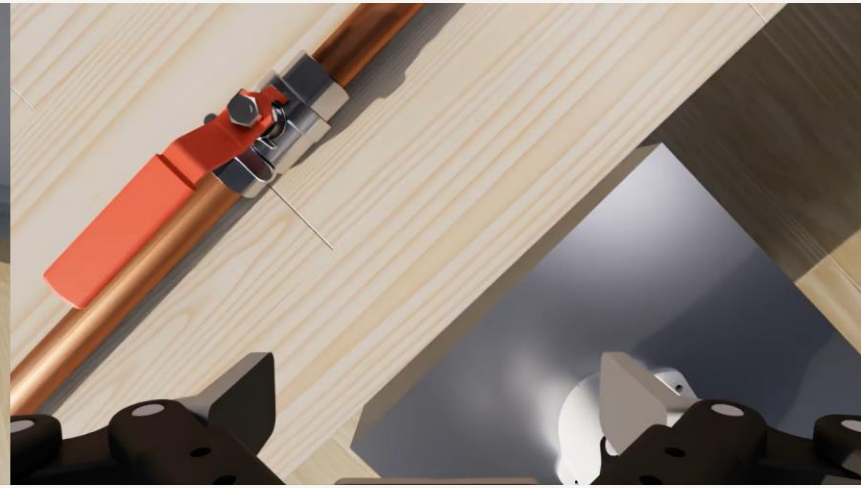
Implementation

Why not use deterministic solutions ?

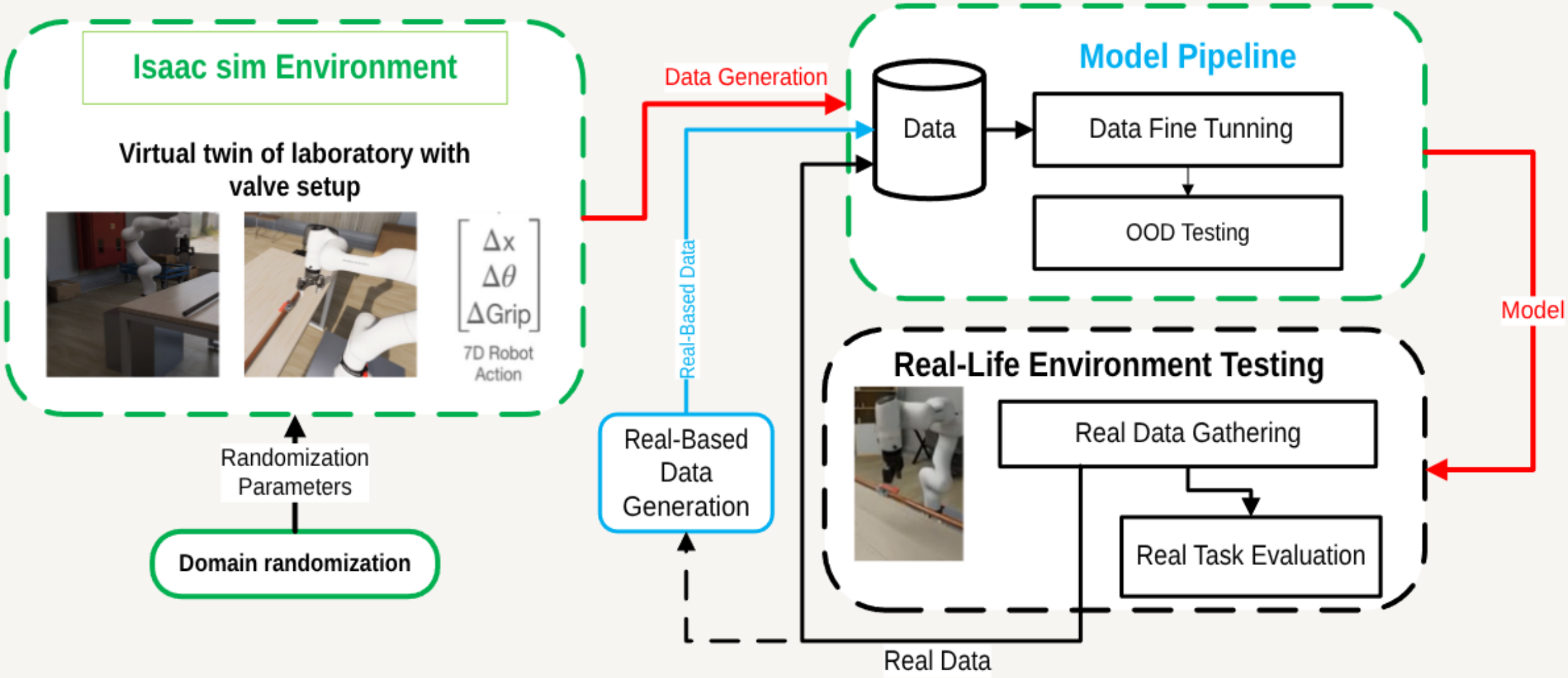
Why not record real data ?



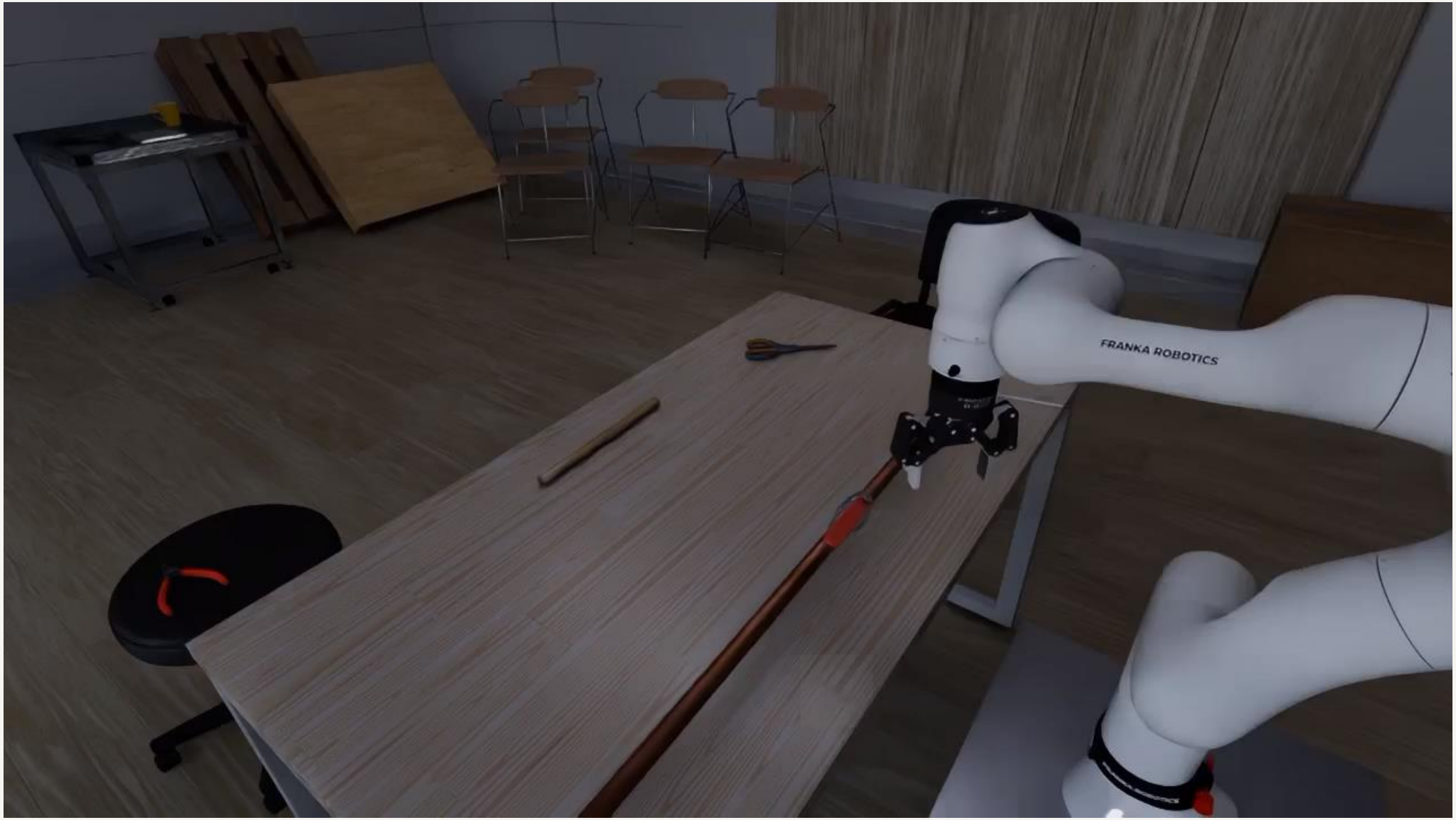




Implementation description

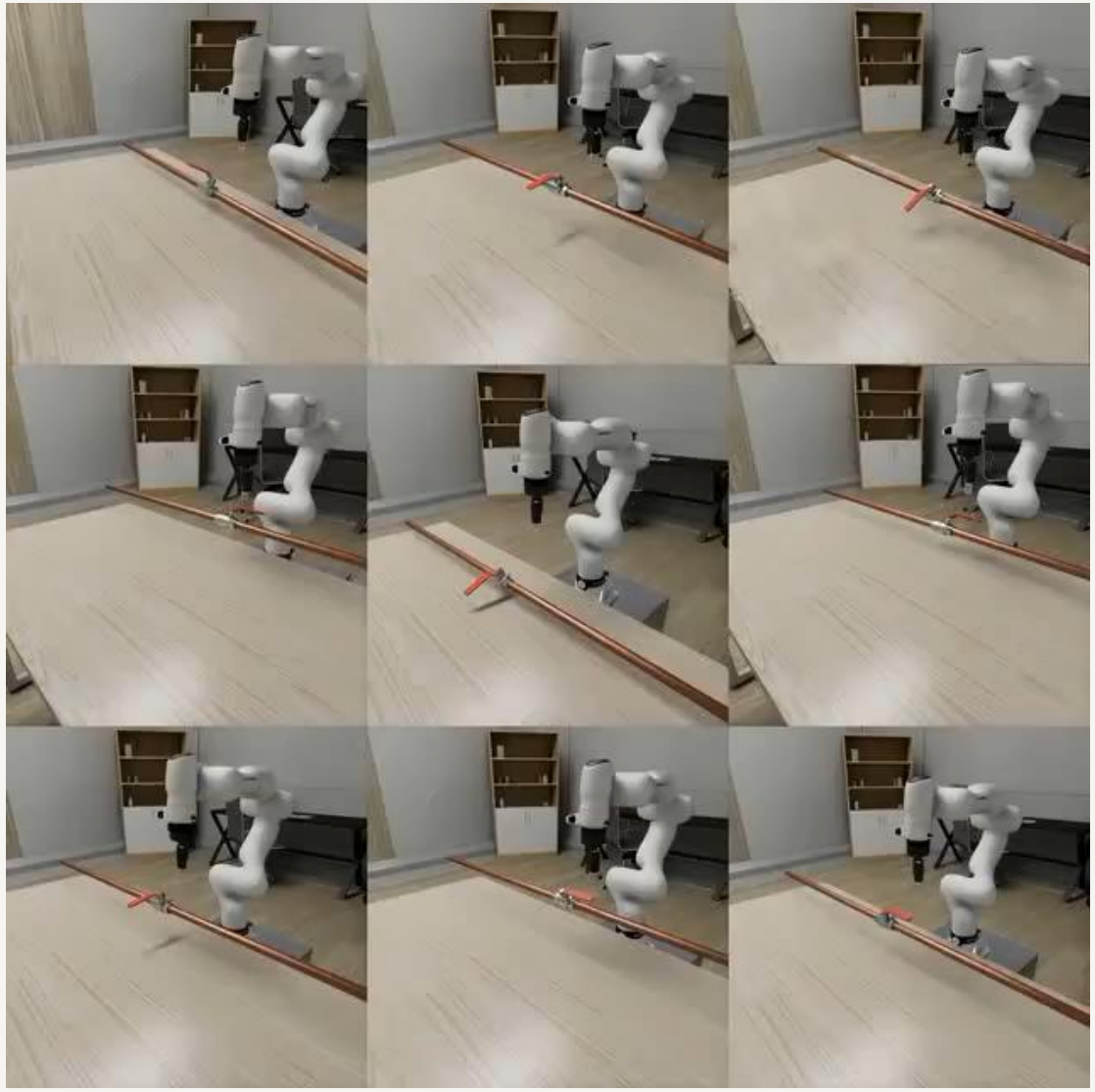


Domain randomization



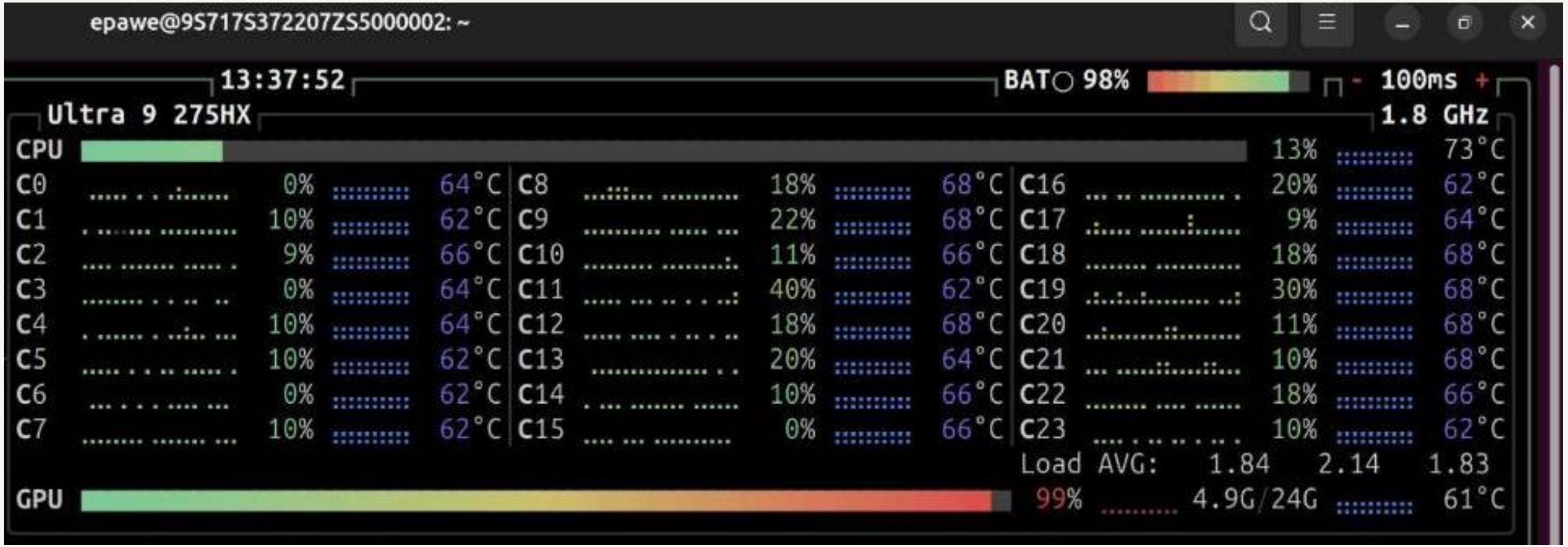
Implementation description





Computational cost

Hardware used:
CPU - Intel® Core™ Ultra 9 Processor 275HX
GPU - RTX5090 32Gb VRAM
RAM – 64 Gb



1 Episode ≈ 22 min

120 Episode = 44 h

Computational cost

Setup	Laptop (RTX 5090 Mobile 24GB + 64GB RAM)	1× NVIDIA A100 (80GB VRAM)	8× RTX 4090 Desktop Cluster (8× 24GB)
LoRA / QLoRA Tuning	~14 –18 hours	~4.5 – 6 hours	~1.0 – 1.3 hours
Full Fine-Tuning	Not Feasible (OOM)	~16 – 22 hours	~2.5 – 3.5 hours

Sources:

https://docs.isaacsim.omniverse.nvidia.com/latest/index.html

https://developer.nvidia.com/blog/enhance-robot-learning-with-synthetic-trajectory-data-generated-by-world-foundation-models/

Few words about explainability





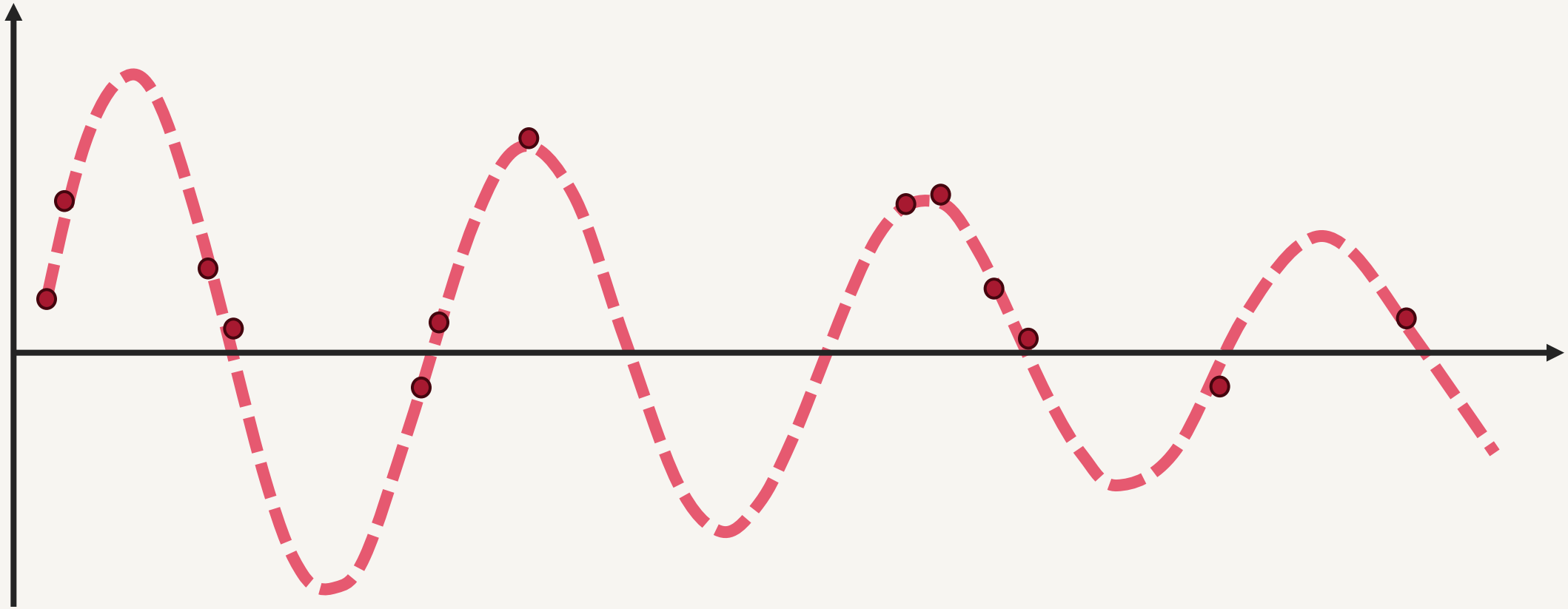
Approach 1: *„Model looks at its answer and shows what parts of the input made it decide”*
And this is biased (but better than nothing)

Approach 2: *Model is built in such a way, the decision is trackable*
(and this usually requires physics-informed structure)

„I believe its a correct solution because it fits data points”

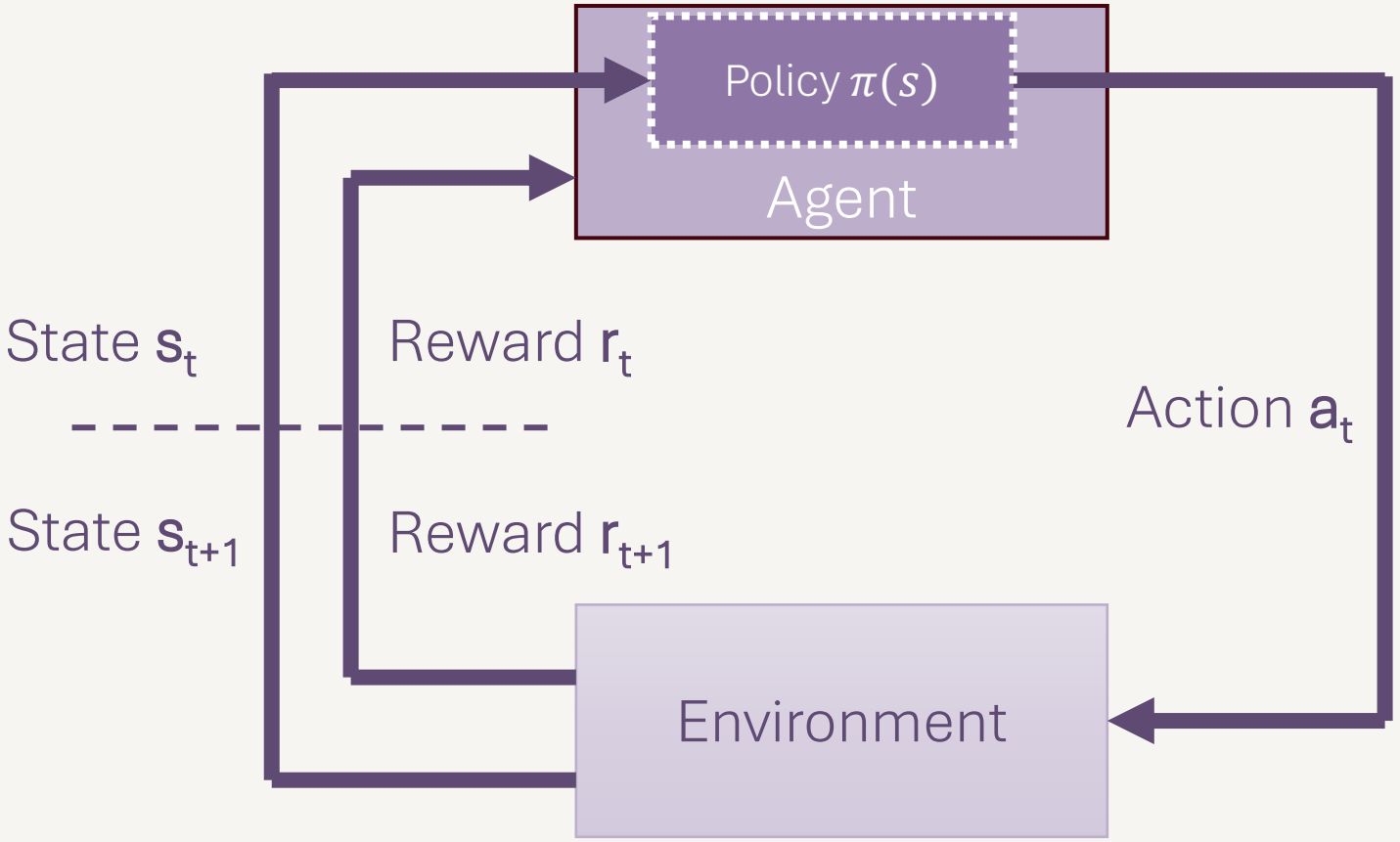
vs

„I believe its a correct solution because it fits underlying physics, the model that I assumed has values of parameters equal to […]”



Better?

Explainable. But not necessarily better…



„Lets make this action because it maximizes expected reward"

vs

„Lets make this action because my physics-informed simulation tells me it will cause the least strain"

I told you, tradeoffs are everywhere!

What do we lose?

The more „moving parts” system has, the more difficult it is to configure

Forced explainability can prevent us from reaching better solutions
(if our understanding of context is not perfect. e.g., Alpha GO)

When we decide which parts of context knowledge we use and which we discard – we can introduce additional constraints and slow down the training