Physics-inspired Neural Networks for building modeling and control

Loris Di Natale [1,2]
Bratislav Svetozarevic [1]
Philipp Heer [1]
Colin N. Jones [2]

[1]Urban Energy Systems Lab, Empa, Switzerland

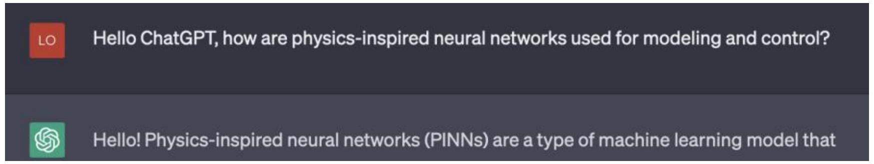
[2]Automatic Control Lab, EPFL, Switzerland

American Control Conference 2023, San Diego

Neural Networks and physical systems

- Neural Networks (NNs) achieve **amazing performance…**
- … but can also **fail spectacularly**
  - They only do ***exactly*** what they are taught

- What about controlling (real-world) **physical systems with NNs**?
  - Data quantity/quality issues
  - Time to converge
  - Exploration phase
  - Guarantees?



Twitter taught Microsoft's AI chatbot to be a racist a**** in less than a day**

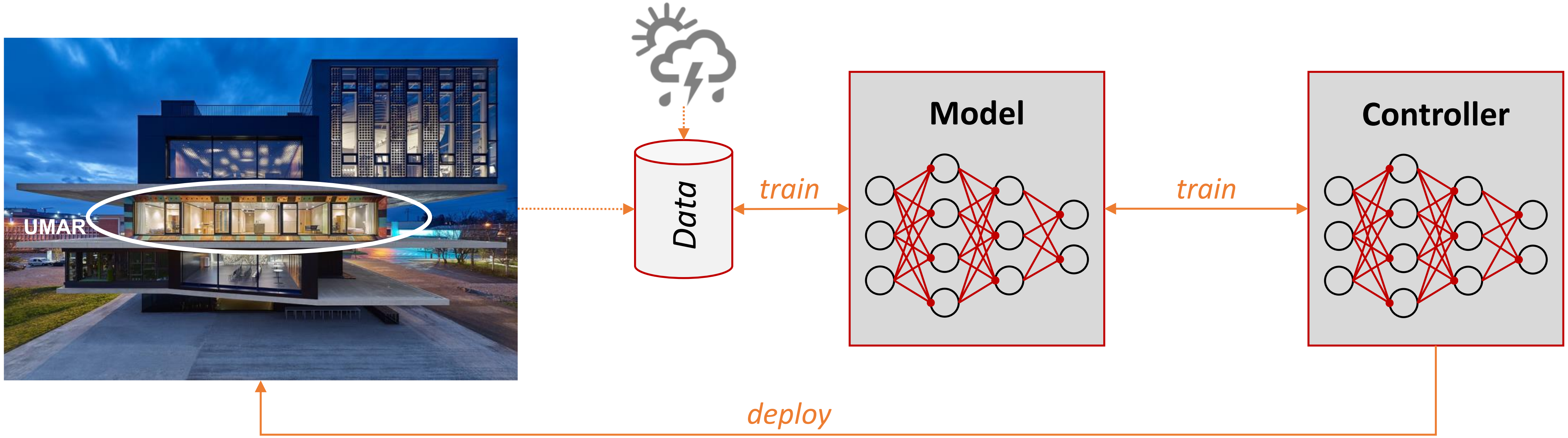
The Guardian

NEST, Empa, Dübendorf Switzerland, © Zooey Braun

Go ***back to simulation***

Neural Networks and physical systems

- Trained **sequentially from data**
- Can choose to use a **NN to model** the system as well

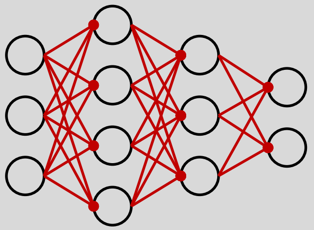


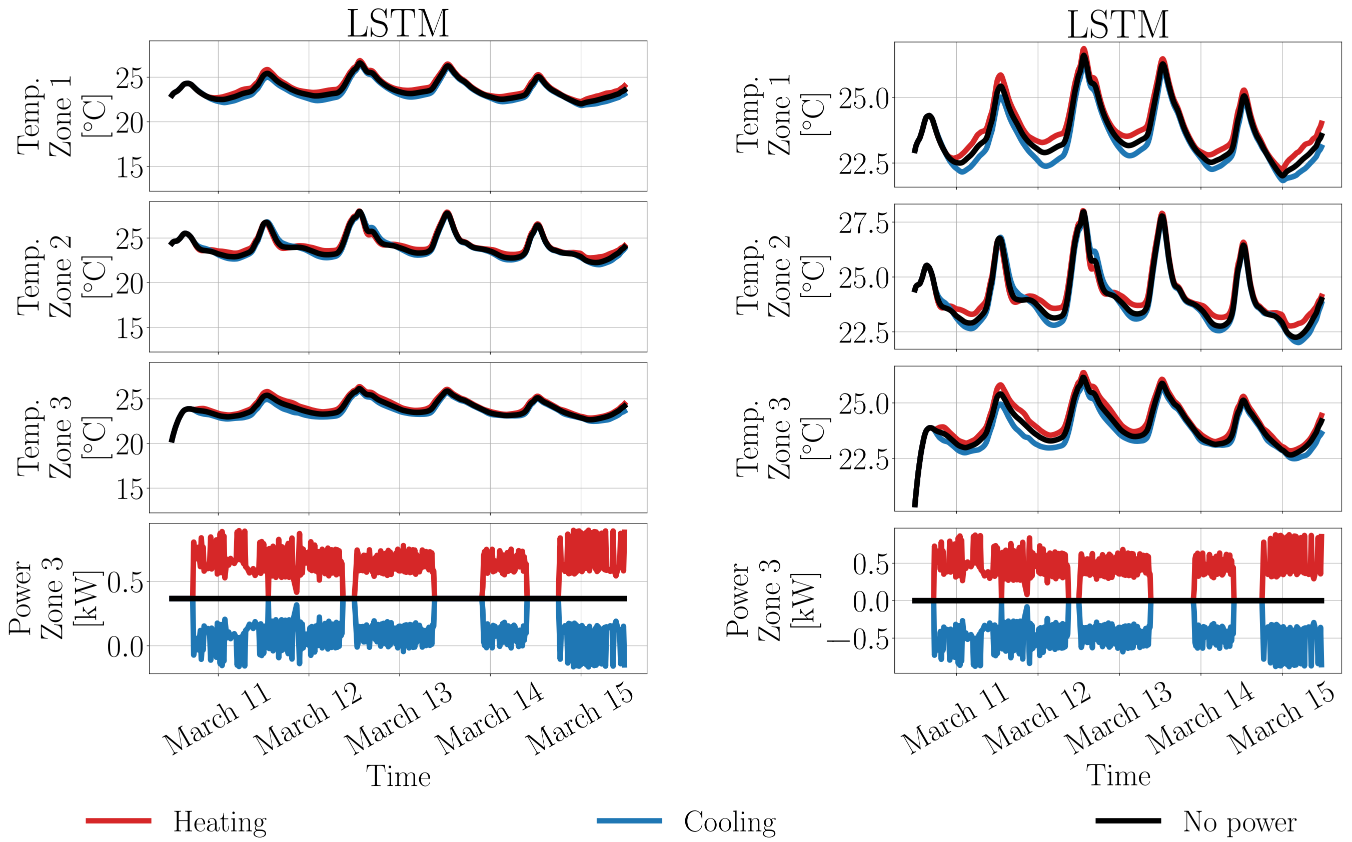
It works! *Can save 25-30% of energy on UMAR compared to industrial baselines,[1]* ***but…***

Neural Networks are physics-agnostic

- They attain **great accuracy**
- They can **violate physical laws** [2,3]

Model





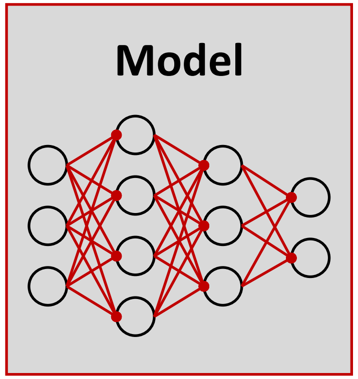
Heating and cooling have almost no ***(or even reversed)*** *impact*

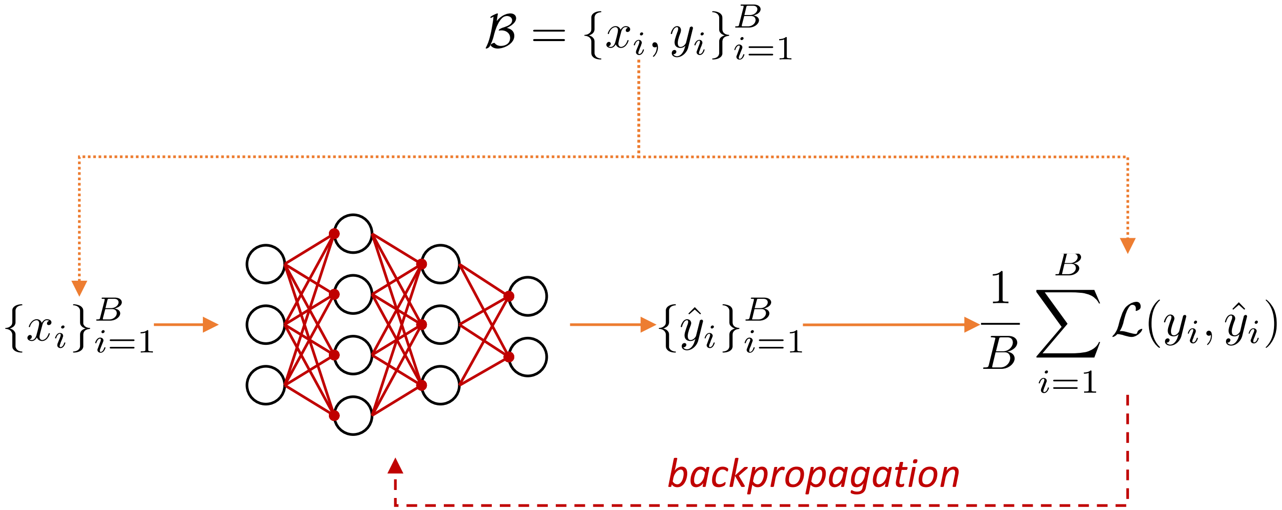
Can we help NNs?

We often have some prior knowledge of the system

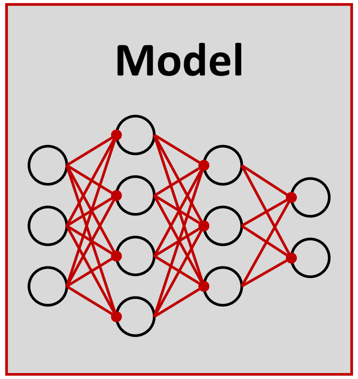
PiNNs for supervised learning

- Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$

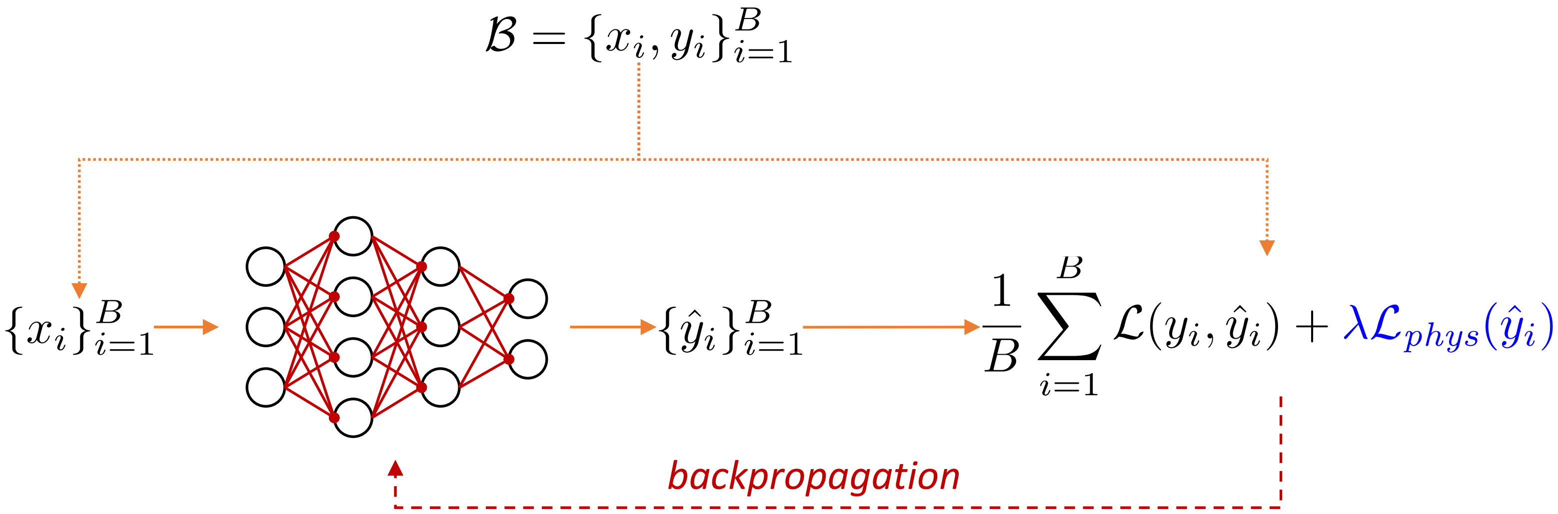




PiNNs for supervised learning

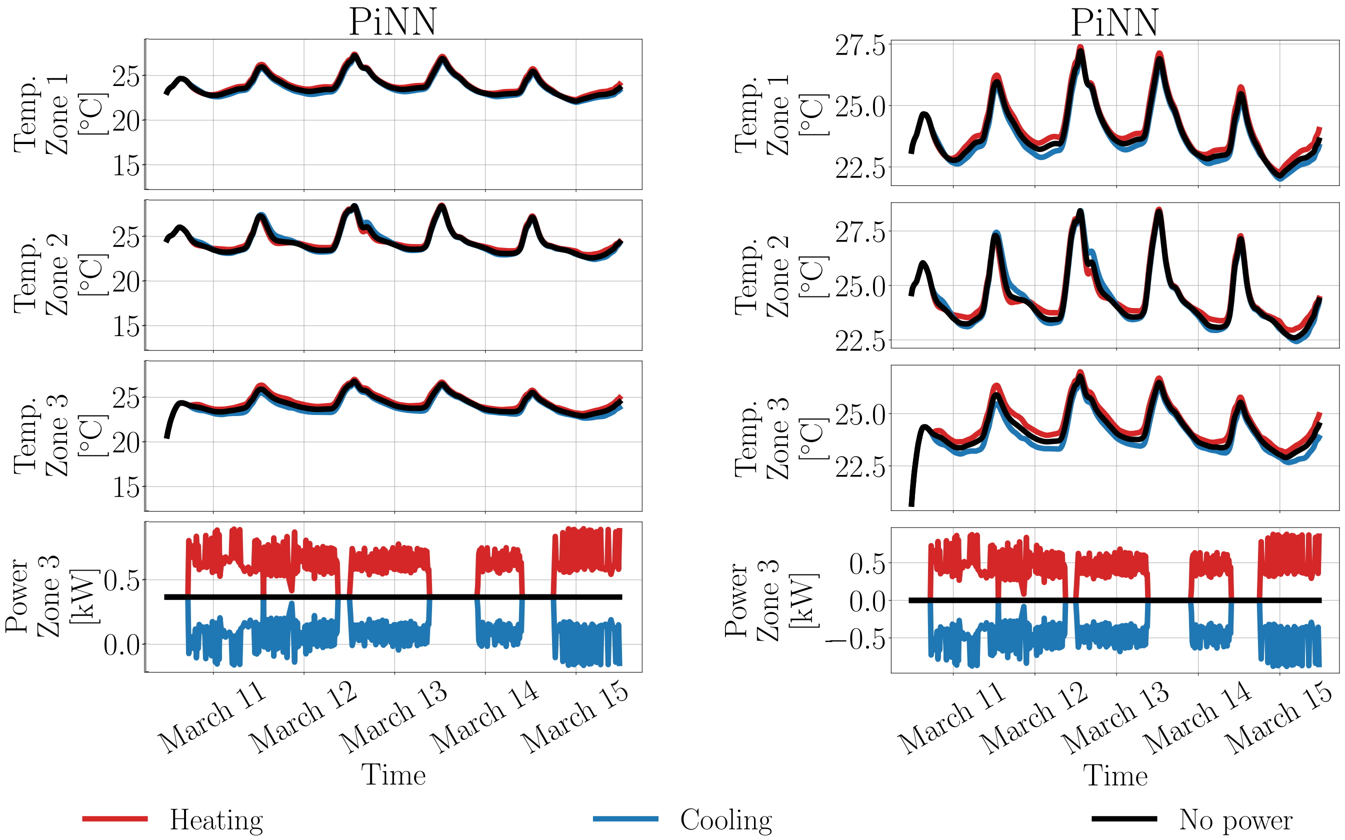


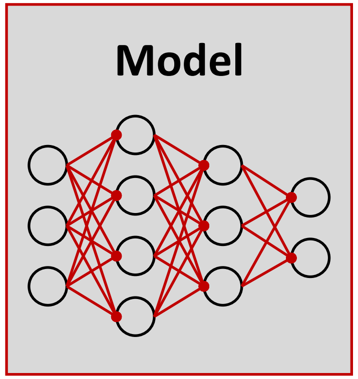
- Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$
- Main idea: **Modified loss function** to steer the NN towards expected solutions
  - Boundary conditions
  - Physical laws (energy/mass/momentum conservation, …)



Do PiNNs work?

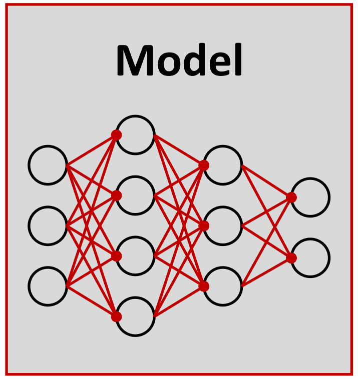
- Want to ensure physical consistency: $\mathcal{L}_{phys}(\hat{y}_i) = \max\left(-\dfrac{\partial \hat{y}_i^{[a]}}{\partial x_i^{[b]}}, 0\right)$ [3]
- **Idea:** heating must heat the building





*Does **not solve** our issues!*

PiNNs for supervised learning



- Modified physical losses are **a good start to help NNs**

Pros

- Ease of implementation
- Can accelerate learning
- Can improve the solution

$$\frac{1}{B}\sum_{i=1}^{B}\mathcal{L}(y_i,\hat{y}_i)+\lambda\mathcal{L}_{phys}(\hat{z}_i,z_i)$$

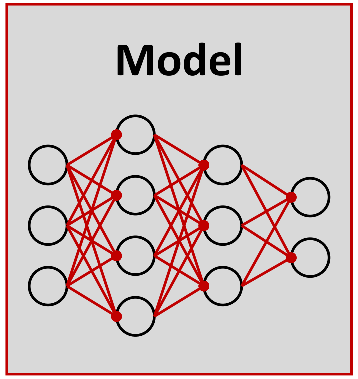
Cons

- Hyperparameter tuning
- **No guarantees**

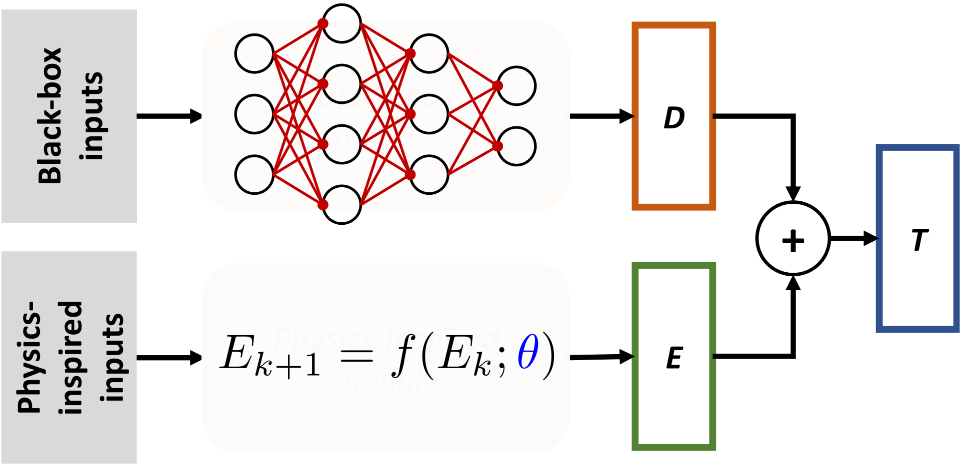
*Whenever possible, you can also try to use **tailored** architectures*

- Hamiltonian NNs,[4] Lagrangian NNs,[5] …

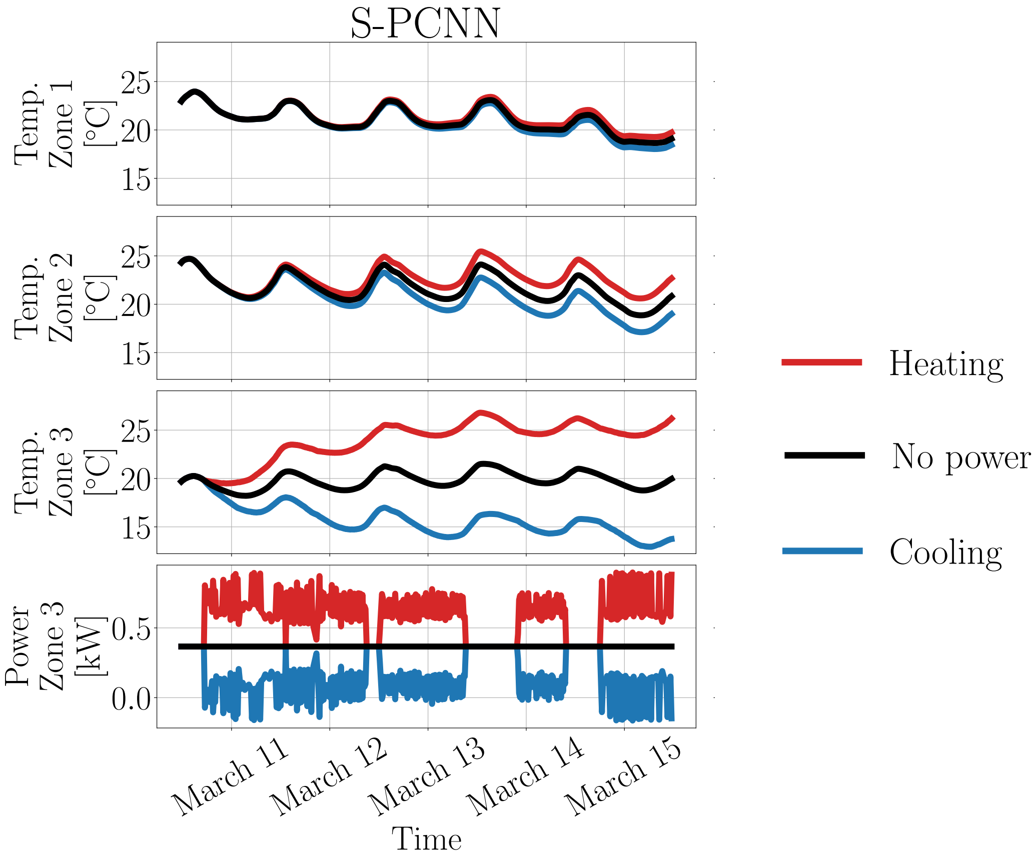
Physically Consistent NNs (PCNNs) [6]



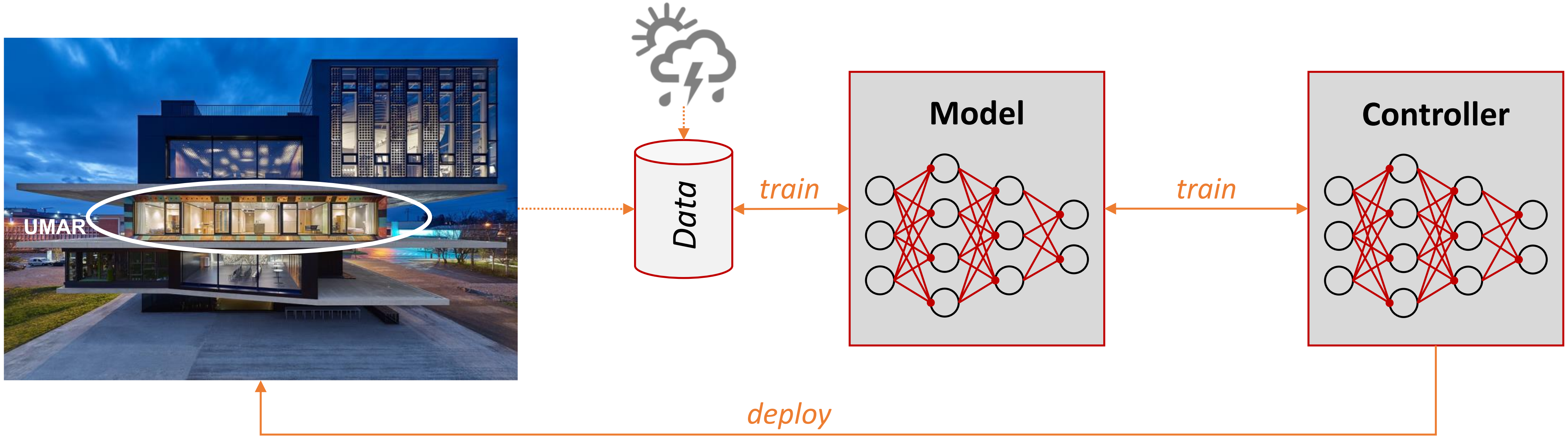
- Let a basic **physics-inspired module** run in parallel of the NN



- **Key:** Train θ **simultaneously** through backpropagation
- **State-of-the-art accuracy** on the analyzed case study
- But **following the laws of physics**

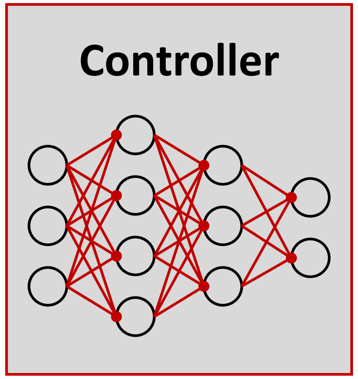


Neural Networks and physical systems

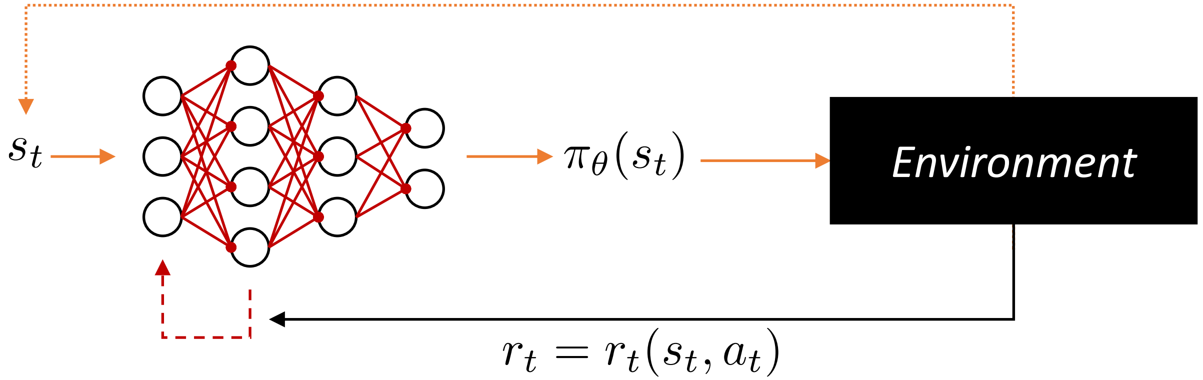


If you have a good model, you can use ***MPC,*** *unless it is impractical*

PiNNs for reinforcement learning

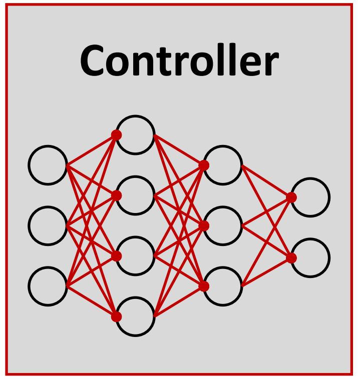


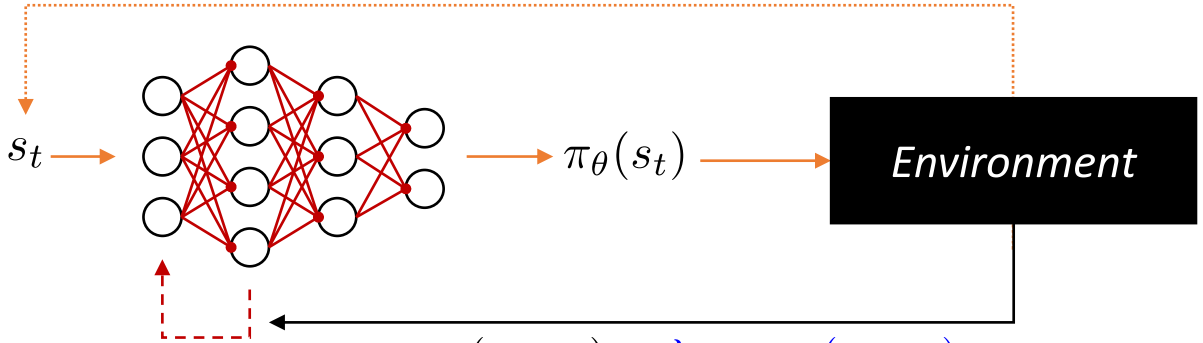
- Learning through interaction with the environment



PiNNs for reinforcement learning

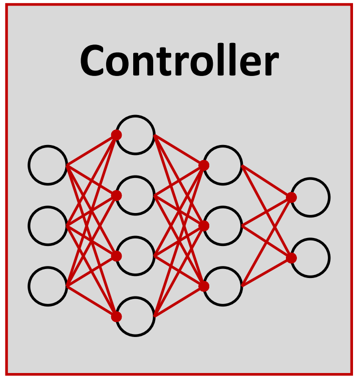
- Learning through interaction with the environment
- **Reward shaping**: modify the feedback to the network (e.g., constraints violations, …)



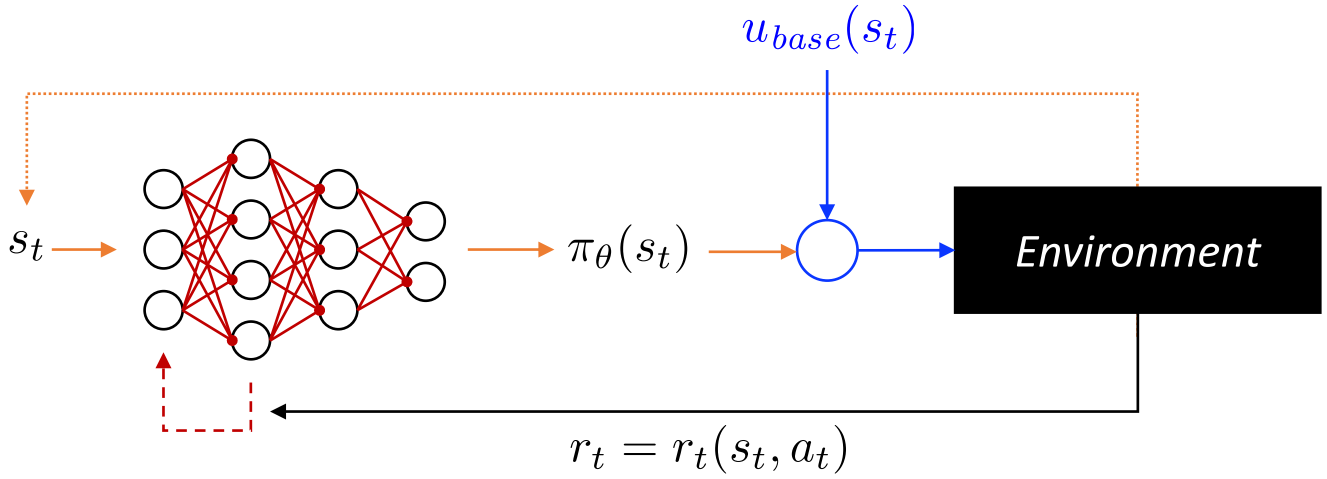


$$r_t = r_t(s_t, a_t) + \lambda r_{phys,t}(s_t, a_t)$$

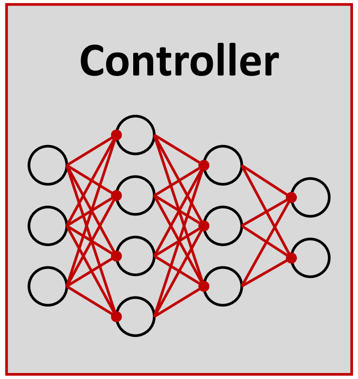
PiNNs for reinforcement learning



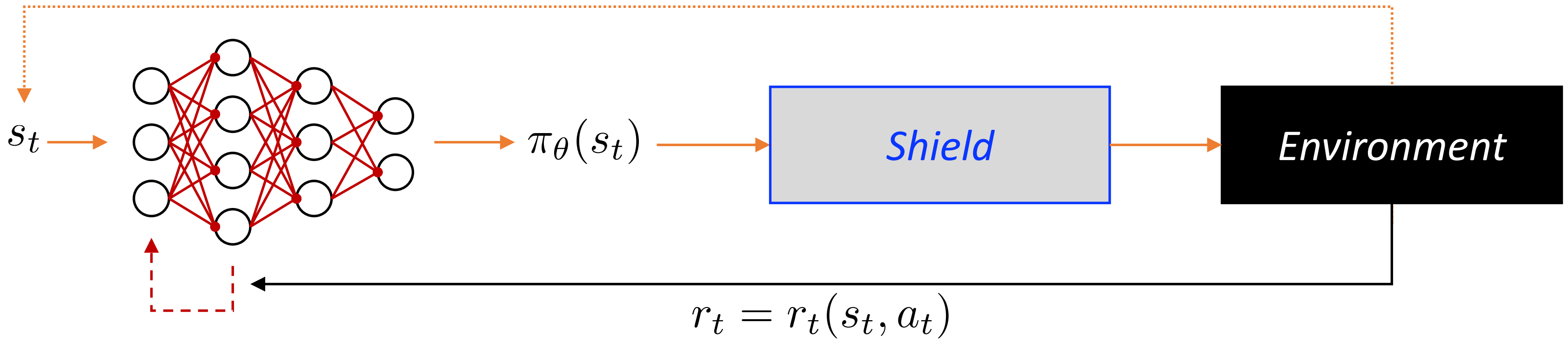
- Learning through interaction with the environment
- **Reward shaping**: modify the feedback to the network (e.g., constraints violations, …)
- **Residual learning:** additive, multiplicative, time varying, …



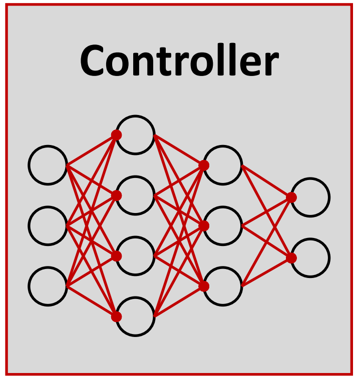
PiNNs for reinforcement learning



- Learning through interaction with the environment
- **Reward shaping**: modify the feedback to the network (e.g., constraints violations, …)
- **Residual learning:** additive, multiplicative, time varying, …
- **Shielding**: Define a filter $\mathcal{C}(s_t)$ to correct the agent when some specifications are not met

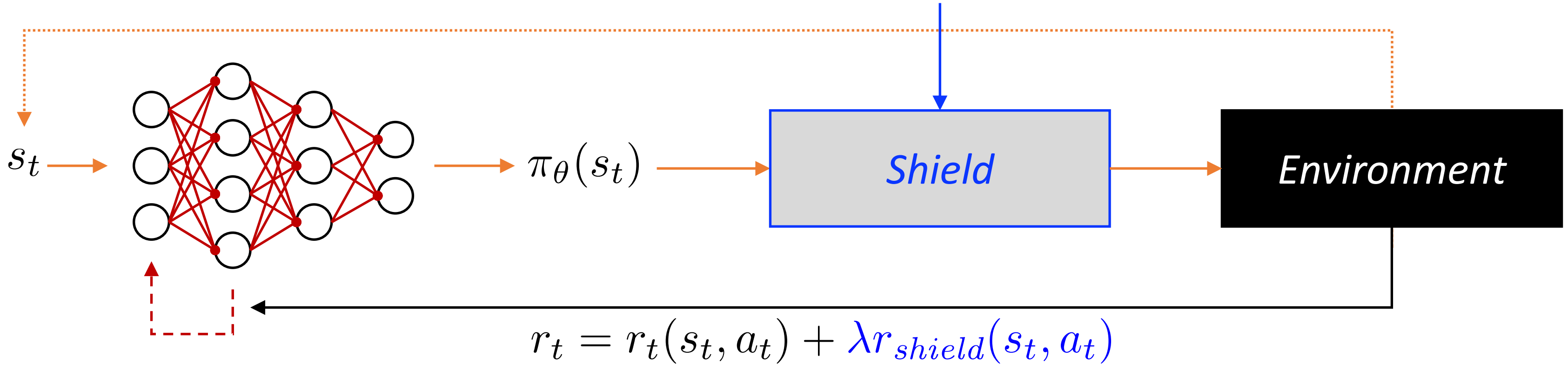


PiNNs for reinforcement learning

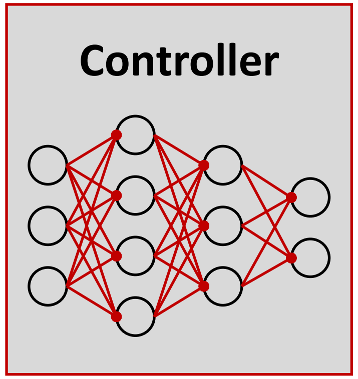


- Learning through interaction with the environment
- **Reward shaping**: modify the feedback to the network (e.g., constraints violations, …)
- **Residual learning:** additive, multiplicative, time varying, …
- **Shielding**: Define a filter $\mathcal{C}(s_t)$ to correct the agent when some specifications are not met
  - Project on the desired set

$$\begin{aligned} \min \quad & ||u - \pi_\theta(s_t)||_2^2 \\ s.t. \quad & u \in \mathcal{C}(s_t) \end{aligned}$$

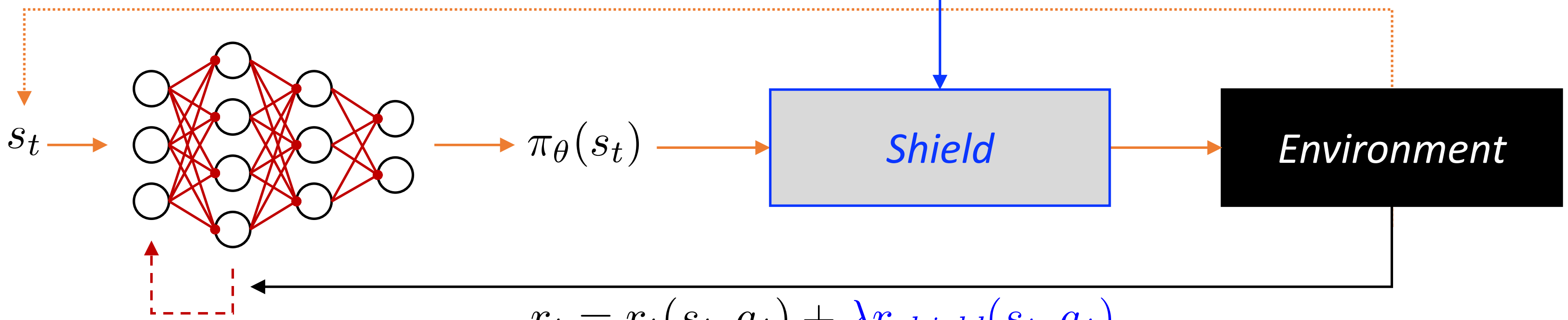


PiNNs for reinforcement learning



- Learning through interaction with the environment
- **Reward shaping**: modify the feedback to the network (e.g., constraints violations, …)
- **Residual learning:** additive, multiplicative, time varying, …
- **Shielding**: Define a filter $\mathcal{C}(s_t)$ to correct the agent when some specifications are not met
  - Project on the desired set
  - Use a known fallback controller $u_{fallback}$

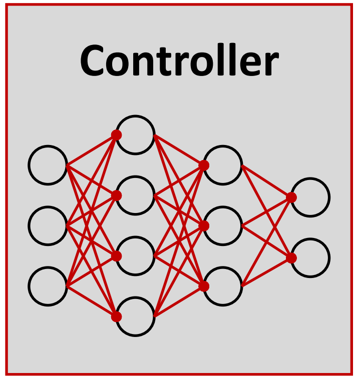
$$u_t = \begin{cases} \pi_\theta(s_t), & \text{if } u \in \mathcal{C}(s_t), \\ u_{fallback}(s_t), & else. \end{cases}$$



$$r_t = r_t(s_t, a_t) + \lambda r_{shield}(s_t, a_t)$$

PiNNs for reinforcement learning

- Introducing physics/knowledge can **accelerate learning** and **lead to better solutions**



*Can we improve the computational efficiency of **shielding** to enforce prior knowledge?*

Computationally Efficient RL [7]

- Use time-varying bounds on the agent's actions to **avoid suboptimal state-action pairs**

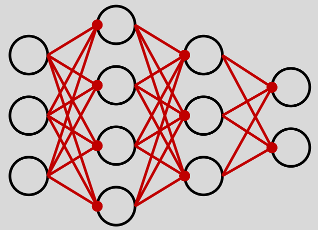
"If it's too cold, then heat"
"If it's too hot, then don't heat"
…

$$\mathcal{C}(s_t) = u^{min}(s_t) \leq u \leq u^{max}(s_t)$$

- The projection is **computationally cheap**: saturate the actions at each step!
- **Modify the gradient update** of the agent so it learns from its mistakes

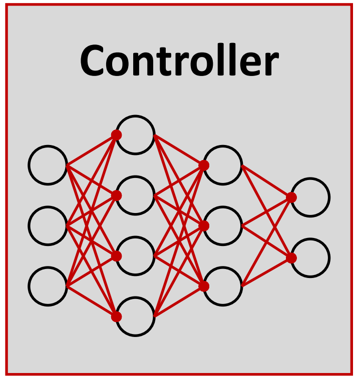
No computational overhead ***but*** *no guarantees*

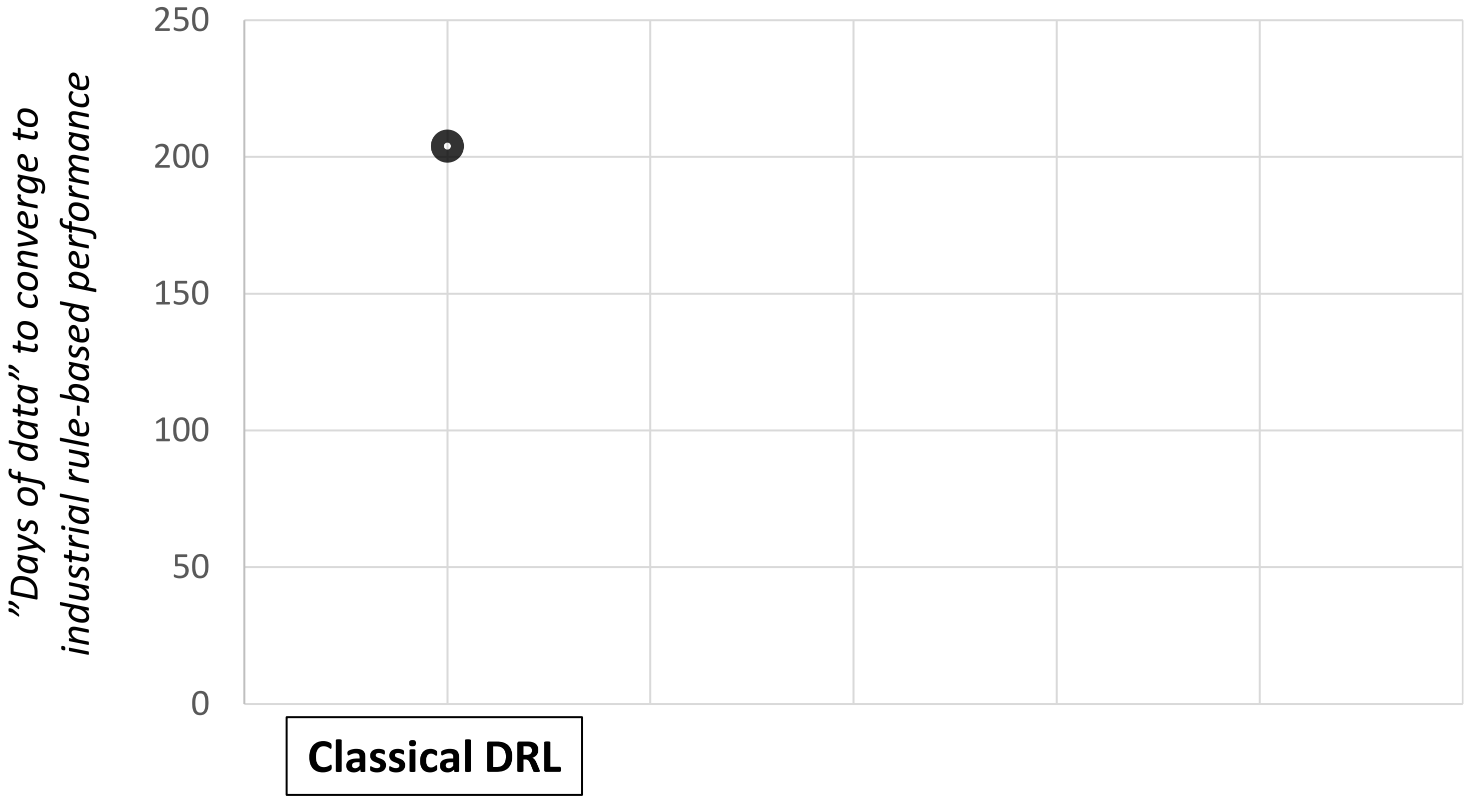
Controller



Computationally Efficient RL [7]

- Use time-varying bounds on the agent's actions to **avoid suboptimal state-action pairs**

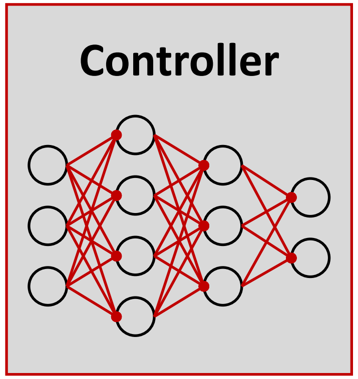


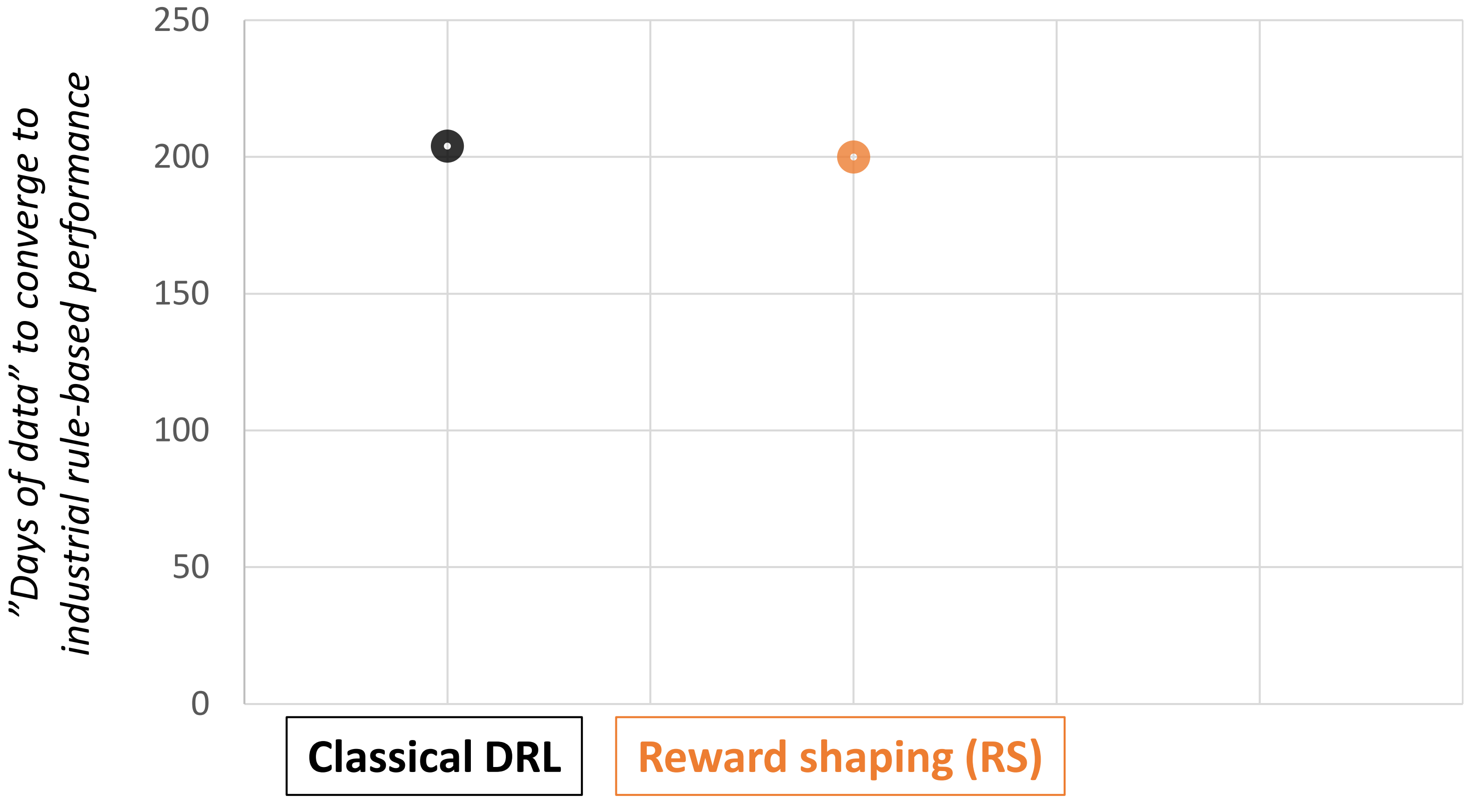


No computational overhead ***but*** *no guarantees*

Computationally Efficient RL [7]

- Use time-varying bounds on the agent's actions to **avoid suboptimal state-action pairs**

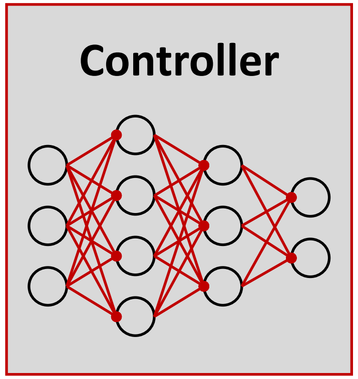


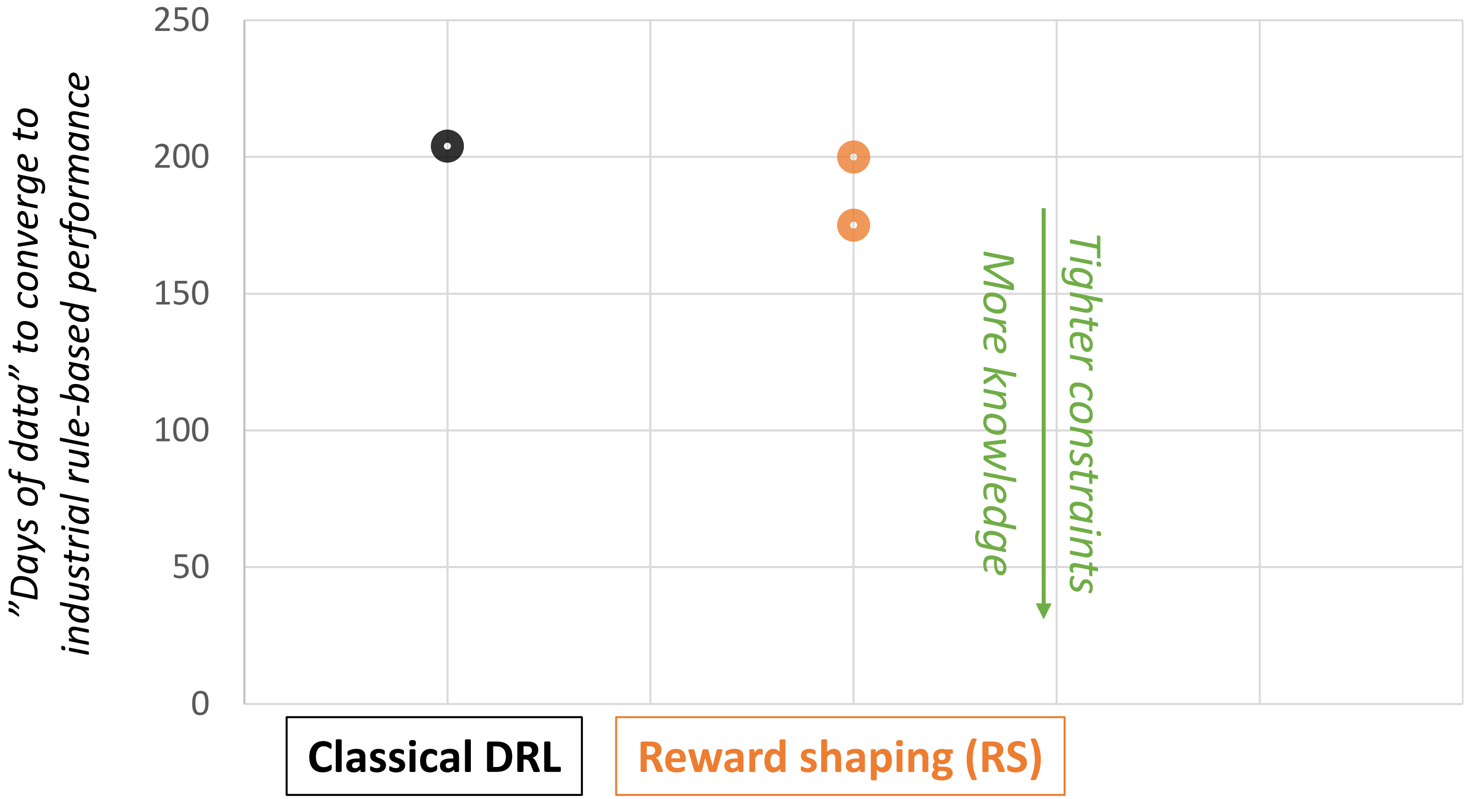


No computational overhead
***but** no guarantees*

Computationally Efficient RL [7]

- Use time-varying bounds on the agent's actions to **avoid suboptimal state-action pairs**

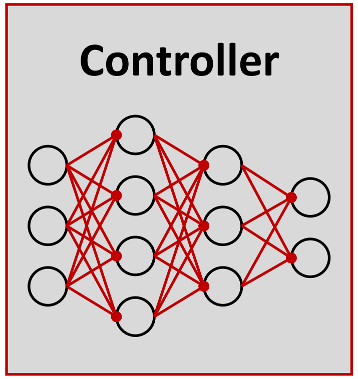


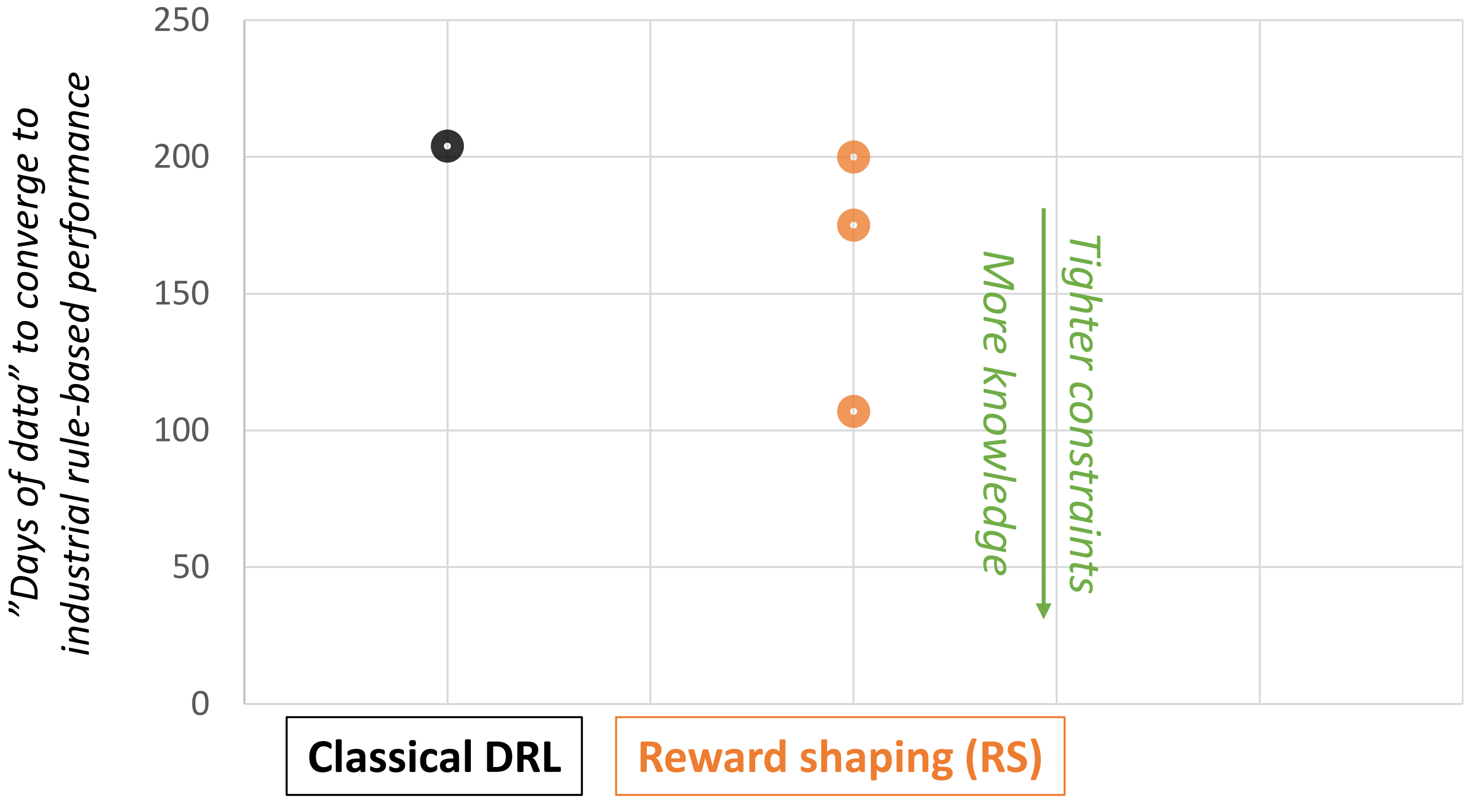


No computational overhead
***but** no guarantees*

Computationally Efficient RL [7]

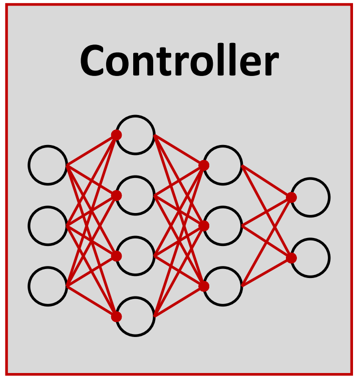
- Use time-varying bounds on the agent's actions to **avoid suboptimal state-action pairs**



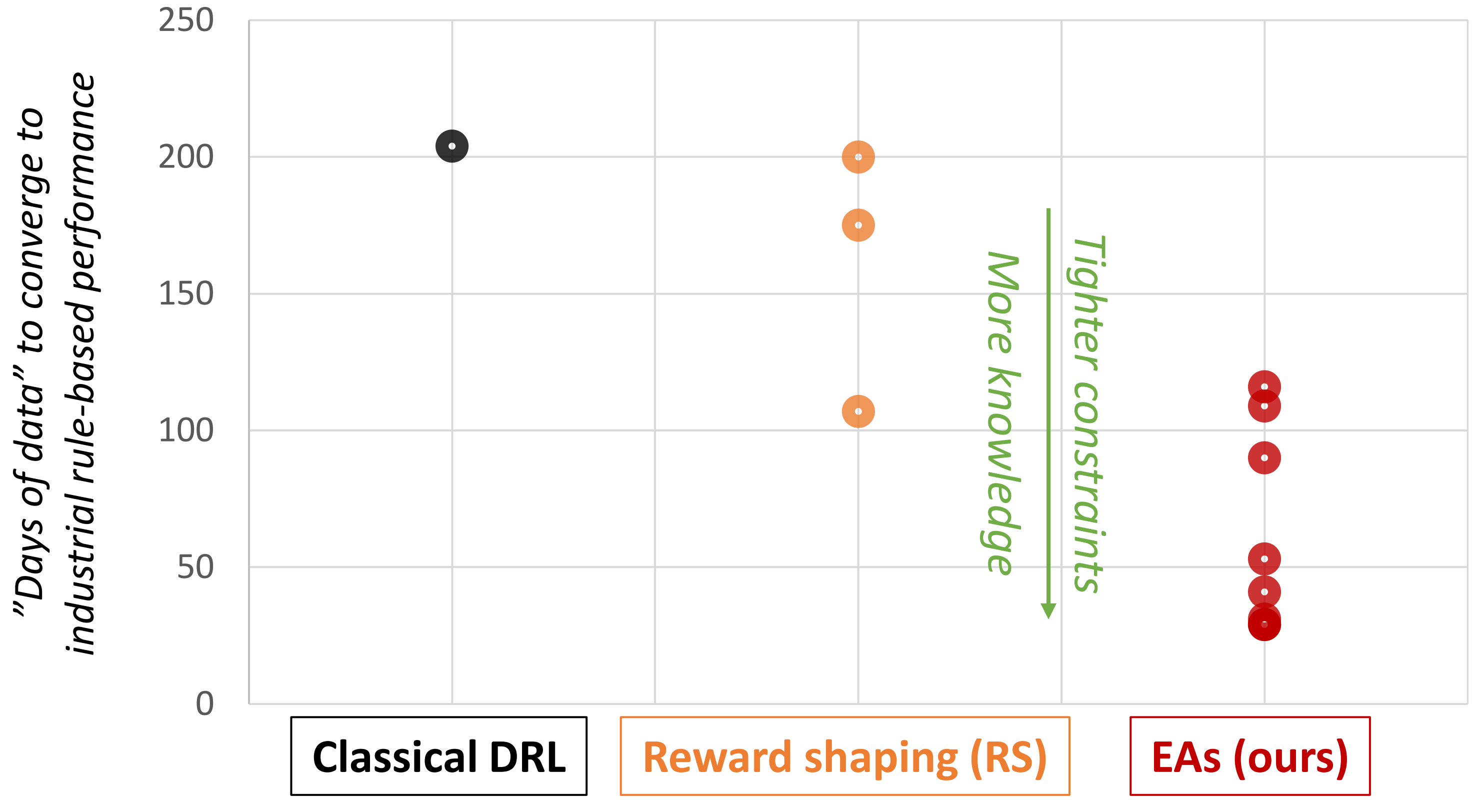


No computational overhead
***but** no guarantees*

Computationally Efficient RL [7]



- Use time-varying bounds on the agent's actions to **avoid suboptimal state-action pairs**



No computational overhead
***but** no guarantees*

*Up to **6-7x** better sample complexity than classical DRL*

*Up to **2-3x** better sample complexity than RS*

Key takeaways

- Neural Networks are powerful but **physics-agnostic**

PiNNs for modeling

You can try different **informative loss functions**

If possible, you can try **tailored architectures** like PCNNs

PiNNs for control

You can try different **reward functions** and **residual learning techniques**

Shielding can give guarantees but is **usually complex**

Thank you for your attention!

Loris DI NATALE
Urban Energy Systems Lab, Empa
Laboratoire d'Automatique, EPFL

loris.dinatale@empa.ch
https://www.empa.ch/web/dilo

References

[1] Di Natale, L., Svetozarevic, B., Heer, P., & Jones, C. N. (2022, June). Near-optimal deep reinforcement learning policies from data for zone temperature control. In *2022 IEEE 17th International Conference on Control & Automation (ICCA)* (pp. 698-703). IEEE.
[2] Geirhos, R., Jacobsen, J. H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, *2*(11), 665-673.
[3] Di Natale, L., Svetozarevic, B., Heer, P., & Jones, C. N. (2023). Towards scalable physically consistent neural networks: An application to data-driven multi-zone thermal building models. *Applied Energy, 340*, 121071.
[4] Greydanus, S., Dzamba, M., & Yosinski, J. (2019). Hamiltonian neural networks. *Advances in neural information processing systems*, *32*.
[5] Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., & Ho, S. (2020). Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*.
[6] Di Natale, L., Svetozarevic, B., Heer, P., & Jones, C. N. (2022). Physically consistent neural networks for building thermal modeling: theory and analysis. *Applied Energy*, *325*, 119806.
[7] Di Natale, L., Svetozarevic, B., Heer, P., & Jones, C. N. (2022). Efficient Reinforcement Learning (ERL): Targeted Exploration Through Action Saturation. *arXiv preprint arXiv:2211.16691*.

Appendix

Key takeaways

- Neural Networks are powerful but **physics-agnostic**

PiNNs for modeling

You can try different **informative loss functions**

If possible, you can try **tailored architectures** like PCNNs

PiNNs for control

You can try different **reward functions** and **residual learning techniques**

Shielding can give guarantees but is **usually complex**

Key takeaways

- Neural Networks are powerful but **physics-agnostic – but solutions exist!**

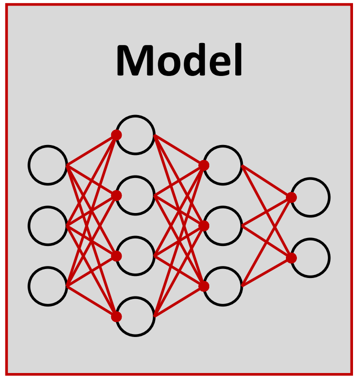
- **PiNNs for modeling** – supervised learning
  - Implementing a physical loss can be **easy and might help** but tuning the weighting factor is not trivial
  - Use **tailored architectures** when possible
    - PCNNs for example attain **state-of-the-art accuracy** for building thermal modeling **while ensuring physical consistency**

- **PiNNs for control** – reinforcement learning
  - While **reward shaping** and **residual learning** are easy, they might not enforce knowledge
  - **Shielding** can preserve prior specifications but is often computationally expensive
    - **Enforcing simple rules instead can already help without computational overhead**

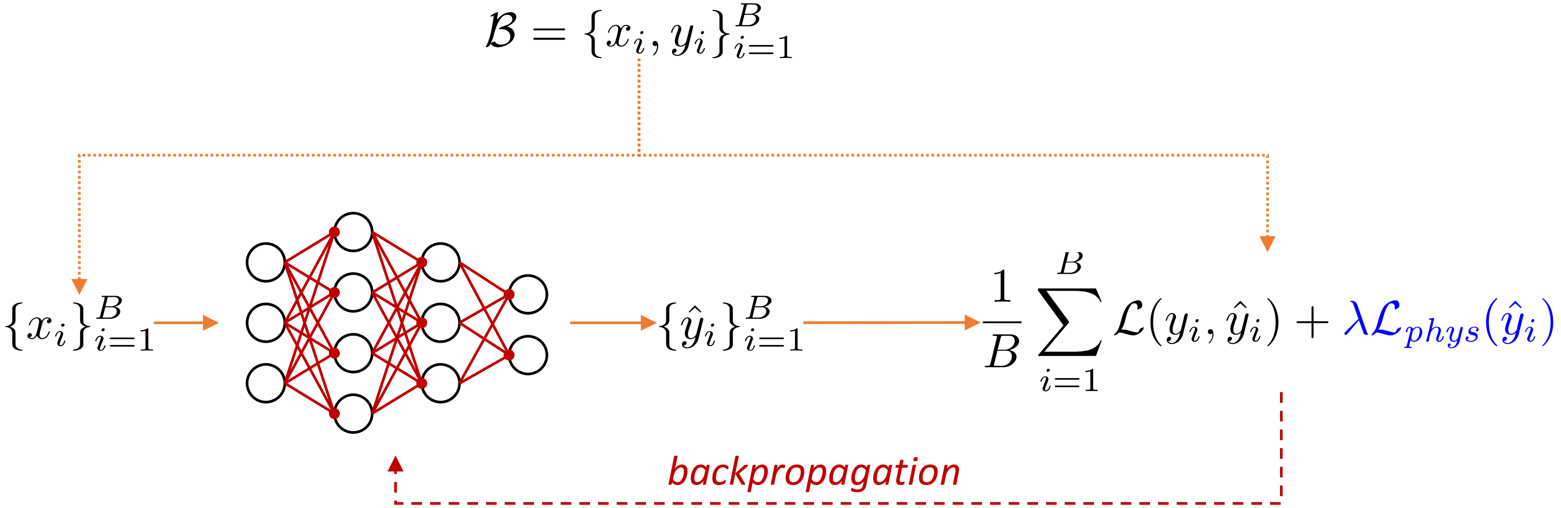
Use (a combination of) these techniques to improve your NNs

- Physics-inspired NNs (PiNNs)

PiNNs for supervised learning

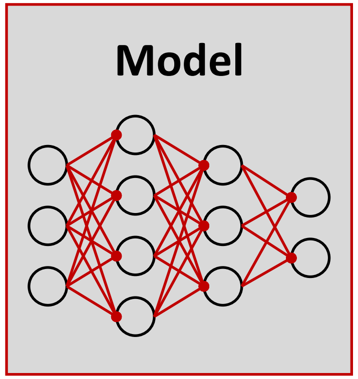


- Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$
- **Main idea**: Modified **loss function** to **steer** the NN towards expected solutions
    - Boundary conditions
    - Physical laws (energy/mass conservation, …)
- Can **augment the NN's output**



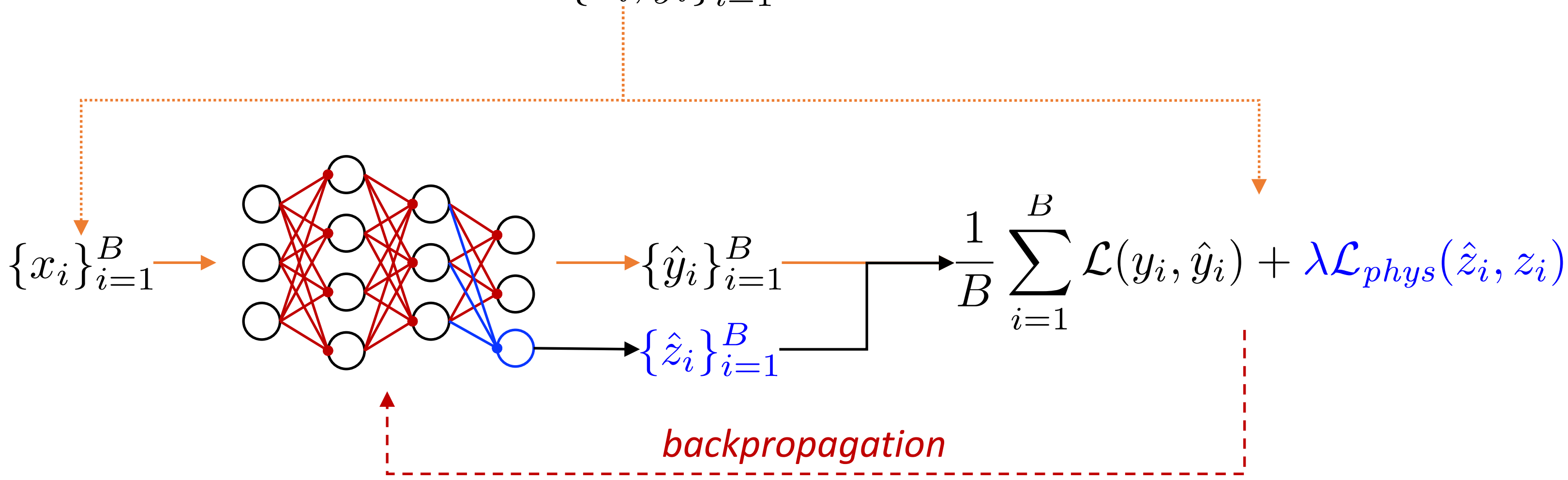
- Physics-inspired NNs (PiNNs)

PiNNs for supervised learning



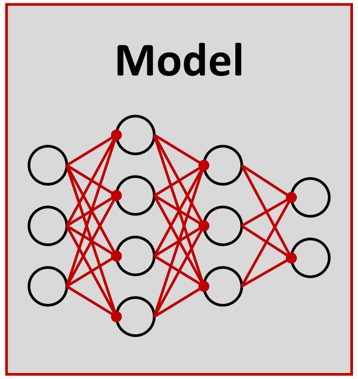
- Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$
- **Main idea**: Modified **loss function** to **steer** the NN towards expected solutions
  - Boundary conditions
  - Physical laws (energy/mass conservation, …)
- Can **augment the NN's output**

$$\mathcal{B} = \{x_i, y_i\}_{i=1}^{B}$$

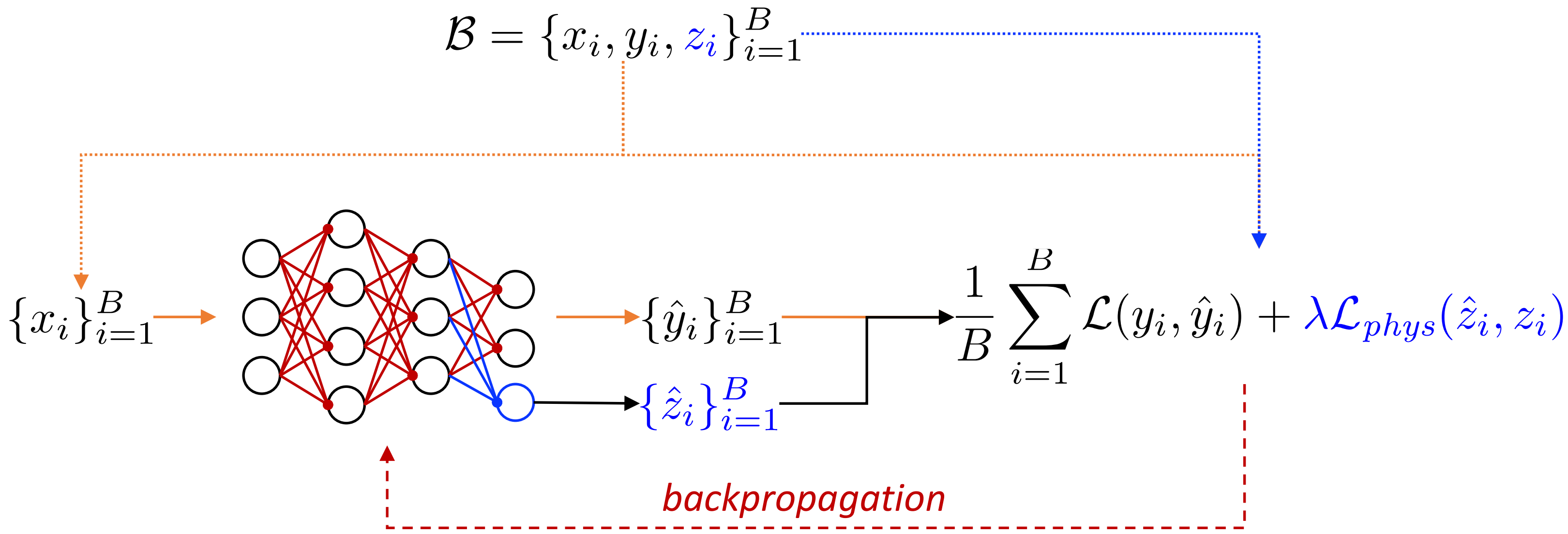


- Physics-inspired NNs (PiNNs)

PiNNs for supervised learning

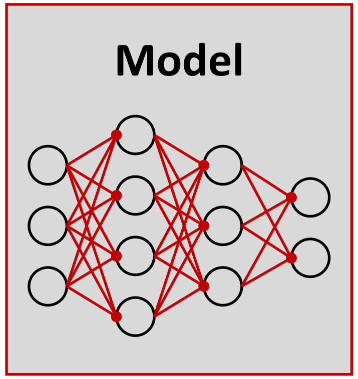


- Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$
- **Main idea**: Modified **loss function** to **steer** the NN towards expected solutions
  - Boundary conditions
  - Physical laws (energy/mass conservation, …)
- Can **augment the NN's output**

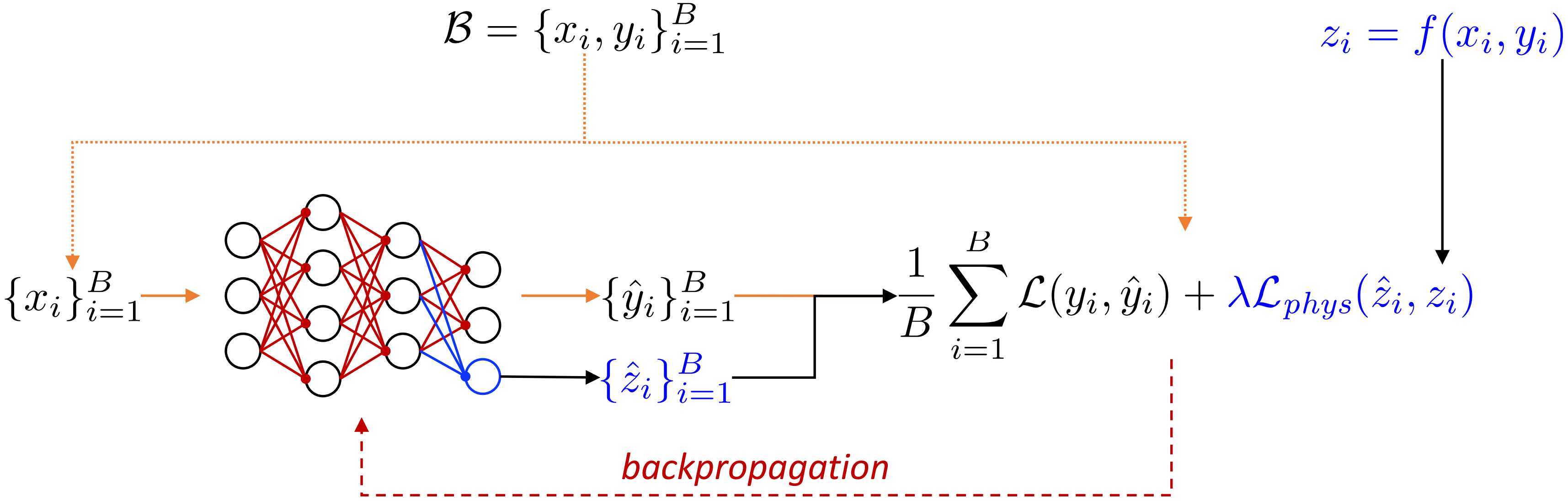


- Physics-inspired NNs (PiNNs)

PiNNs for supervised learning



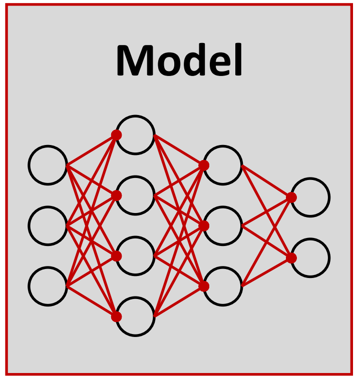
- Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$
- **Main idea**: Modified **loss function** to **steer** the NN towards expected solutions
  - Boundary conditions
  - Physical laws (energy/mass conservation, …)
- Can **augment the NN's output**
- Can **create a bottleneck**

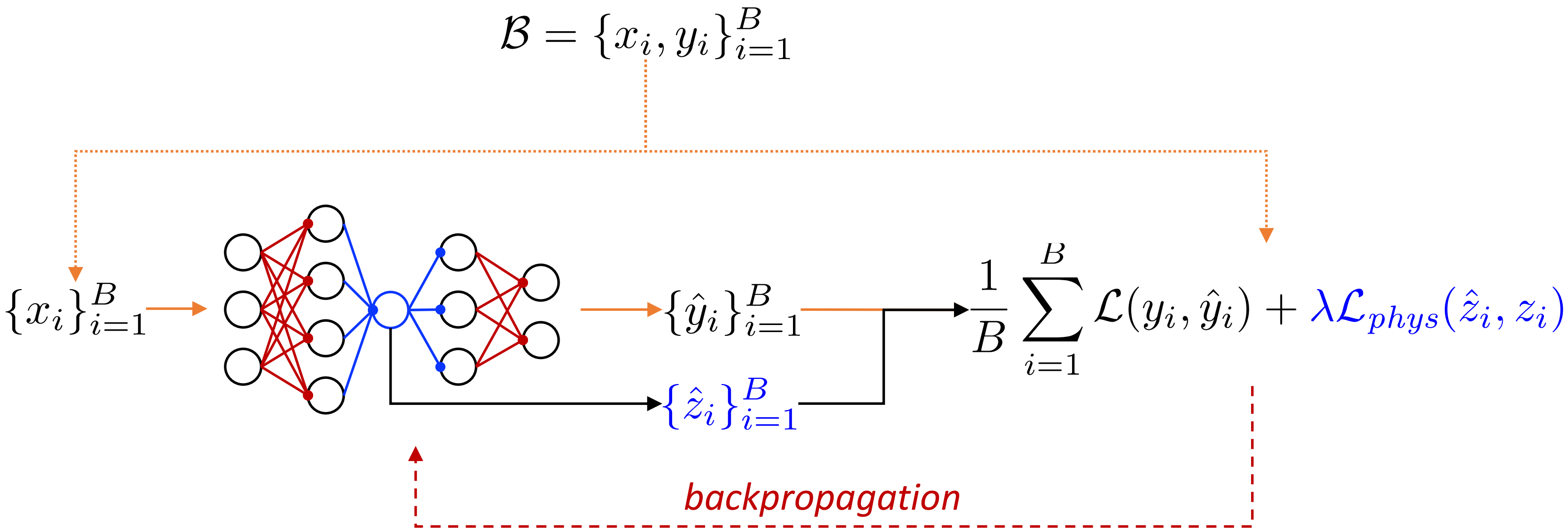


- Physics-inspired NNs (PiNNs)

PiNNs for supervised learning

- Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{N}$
- **Main idea**: Modified **loss function** to **steer** the NN towards expected solutions
  - Boundary conditions
  - Physical laws (energy/mass conservation, …)
- Can **augment the NN's output**
- Can **create a bottleneck**
- …

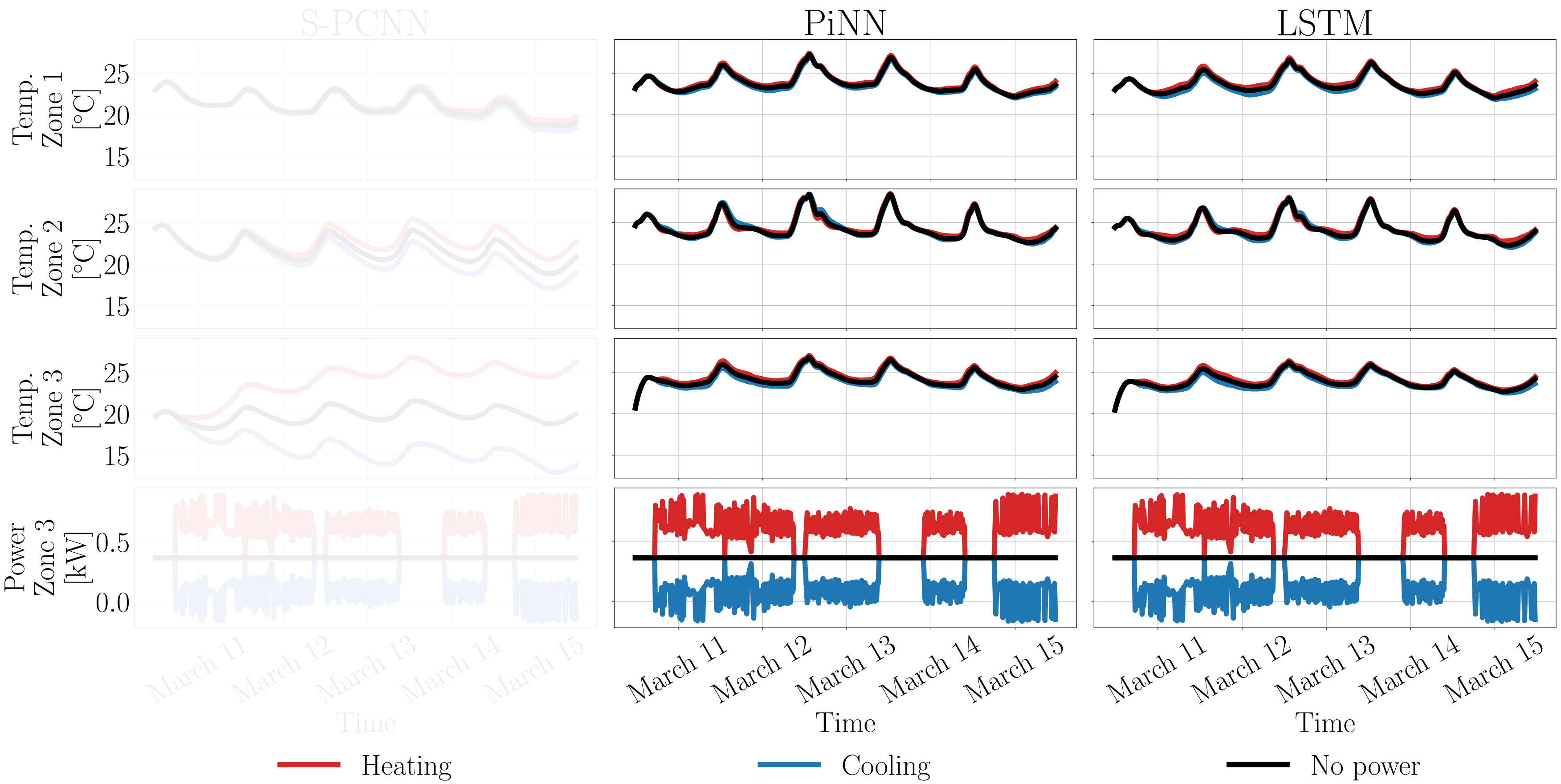




The issue of *shortcut* learning

- Case study on UMAR: learn the **thermodynamics of buildings**
- PiNNs and LSTMs have a **great accuracy** on the **validation data**

Issue of *shortcut learning*
No impact of heating/cooling!

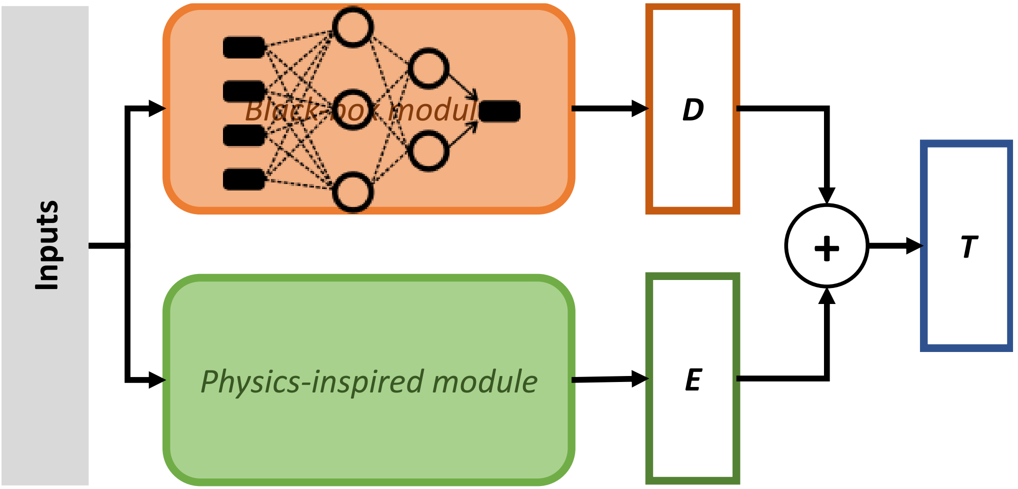


Physically Consistent NNs (PCNNs)

- **Enforce physical properties *by design*** in a module running in parallel of the NN

Building thermodynamics example
We want to ensure:
- **The predicted temperatures increase** when the applied heating power or the outside temperature increases;
- **Physically meaningful energy flows** between the thermal zones.

This **is hard-encoded in the physics-inspired module**, which processes the power inputs, ambient temperatures, and energy exchanges between the zones.
- The parameters are **learned from data**

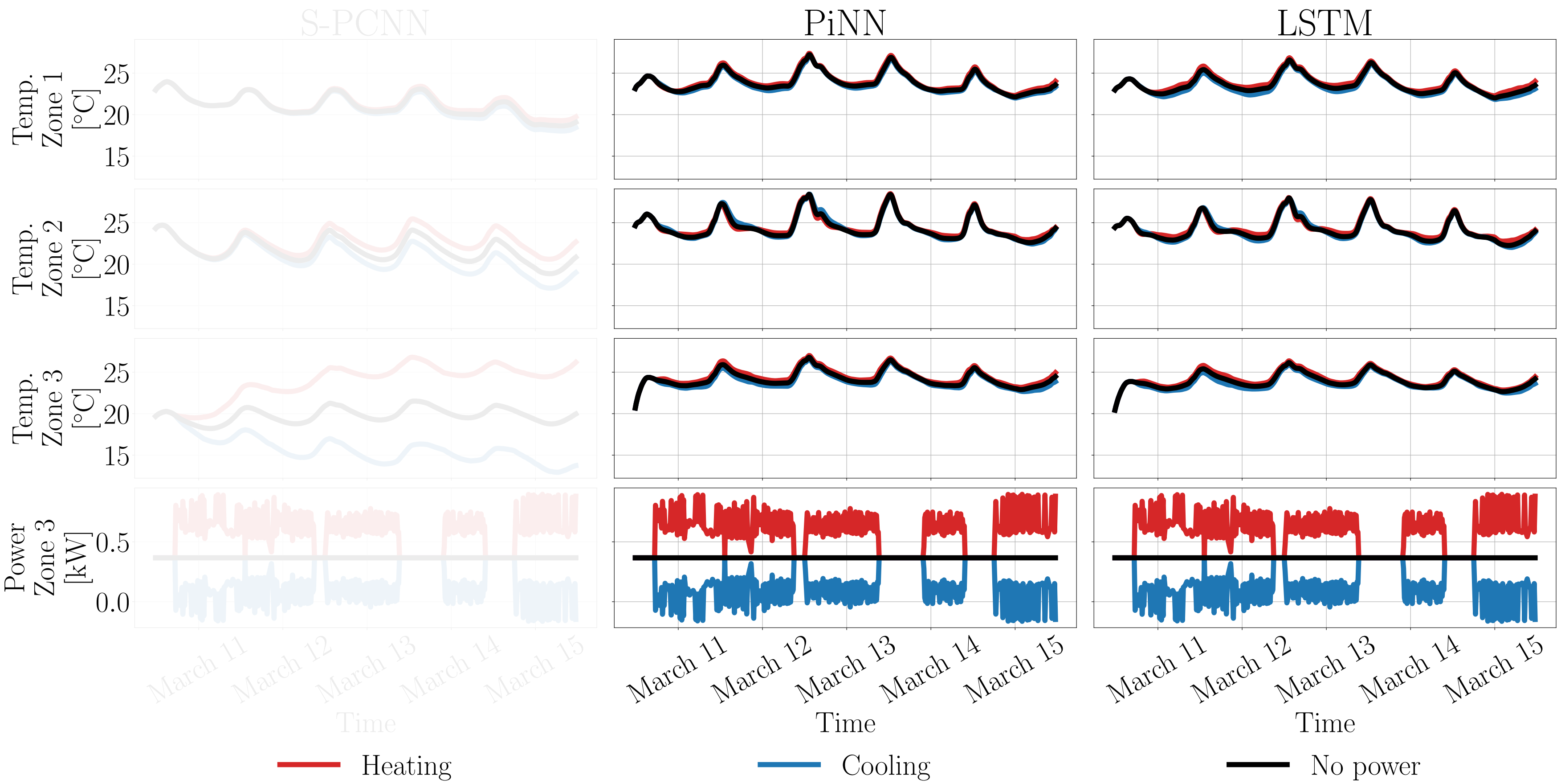


$$
\begin{aligned}
D_{k+1} &= D_k + f(x_k, D_k), \\
E_{k+1} &= E_k + a_h \max\{u_k, 0\} + a_c \min\{u_k, 0\} \\
&\quad - b(T_k - T_k^{out}) - \sum_{z' \in \mathcal{N}(z)} c_{z'}(T_k - T_k^{z'}), \\
T_{k+1} &= D_{k+1} + E_{k+1},
\end{aligned}
\qquad
\left\{
\begin{aligned}
&\frac{\partial T^z_{k+i}}{\partial u^z_{k+j}} > 0 \\
{}^*\; &\frac{\partial T^z_{k+i}}{\partial T^{out}_{k+j}} > 0 \\
&\frac{\partial T^z_{k+i}}{\partial T^y_{k+j}} > 0
\end{aligned}
\right.
$$

* under mild assumptions on a_h, a_c, b, c

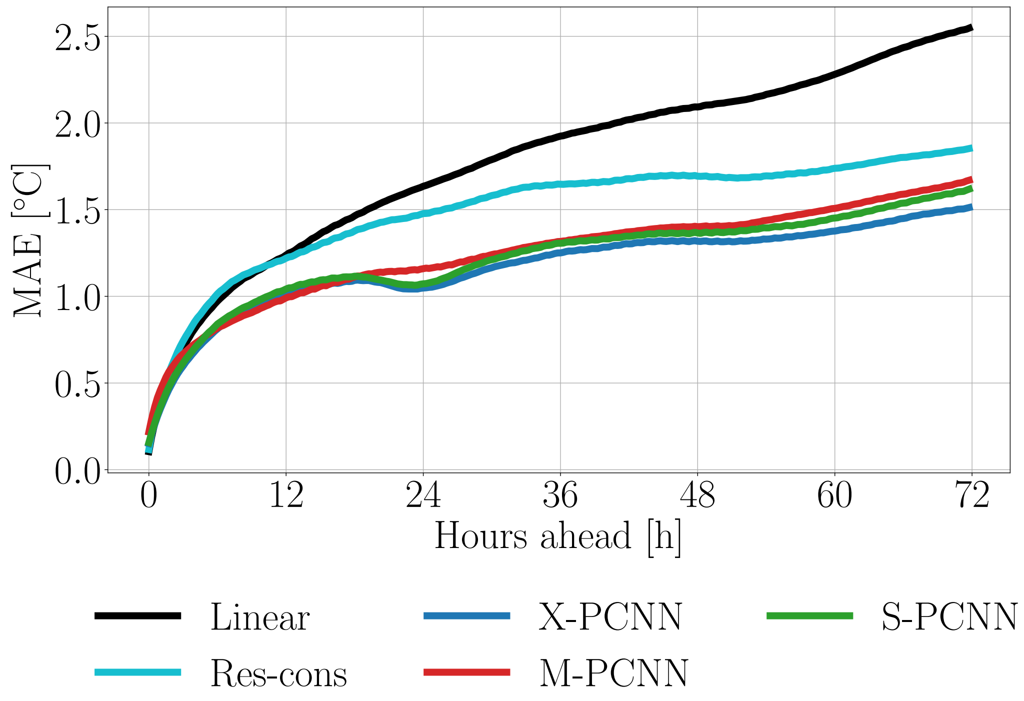
Solving the *shortcut* learning issue

- Case study on UMAR: learn the **thermodynamics of buildings**
- PiNNs and LSTMs have a **great accuracy** on the **validation data** **– PCNNs are physically consistent *by construction***



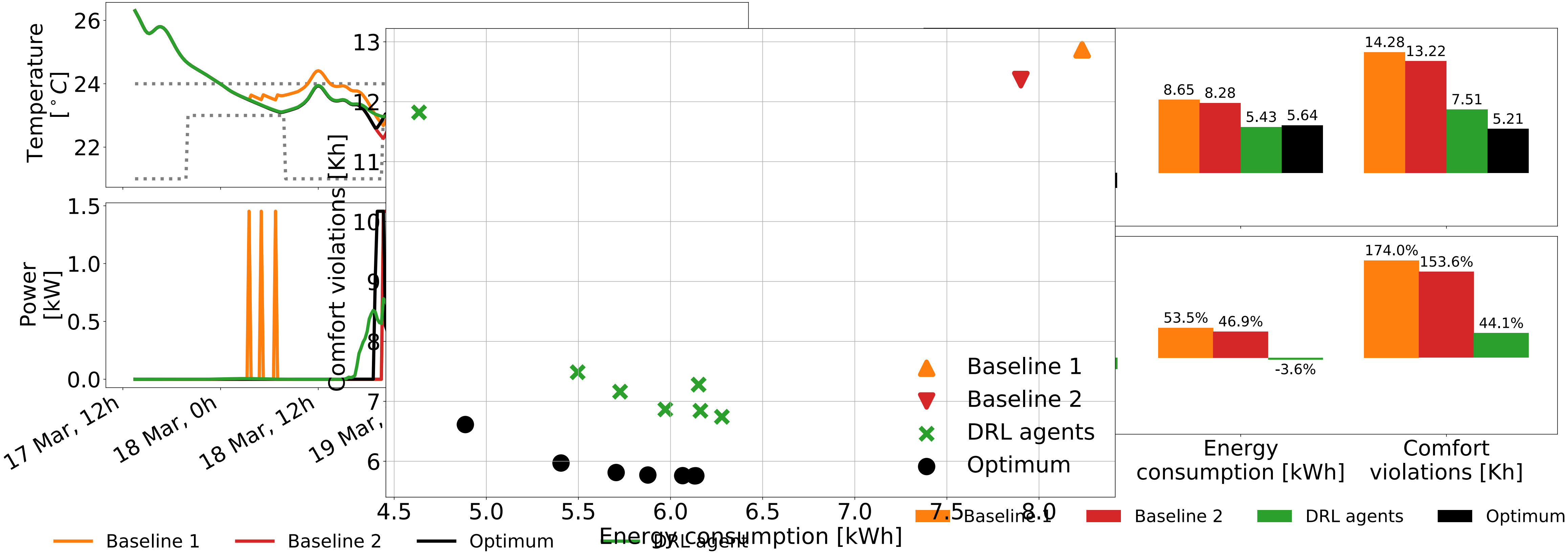
Solving the *shortcut* learning issue

- Case study on UMAR: learn the **thermodynamics of buildings**
- PiNNs and LSTMs have a **great accuracy** on the **validation data – PCNNs are physically consistent *by construction***
- But they also **outperform all the other physically consistent methods by 17-35%**



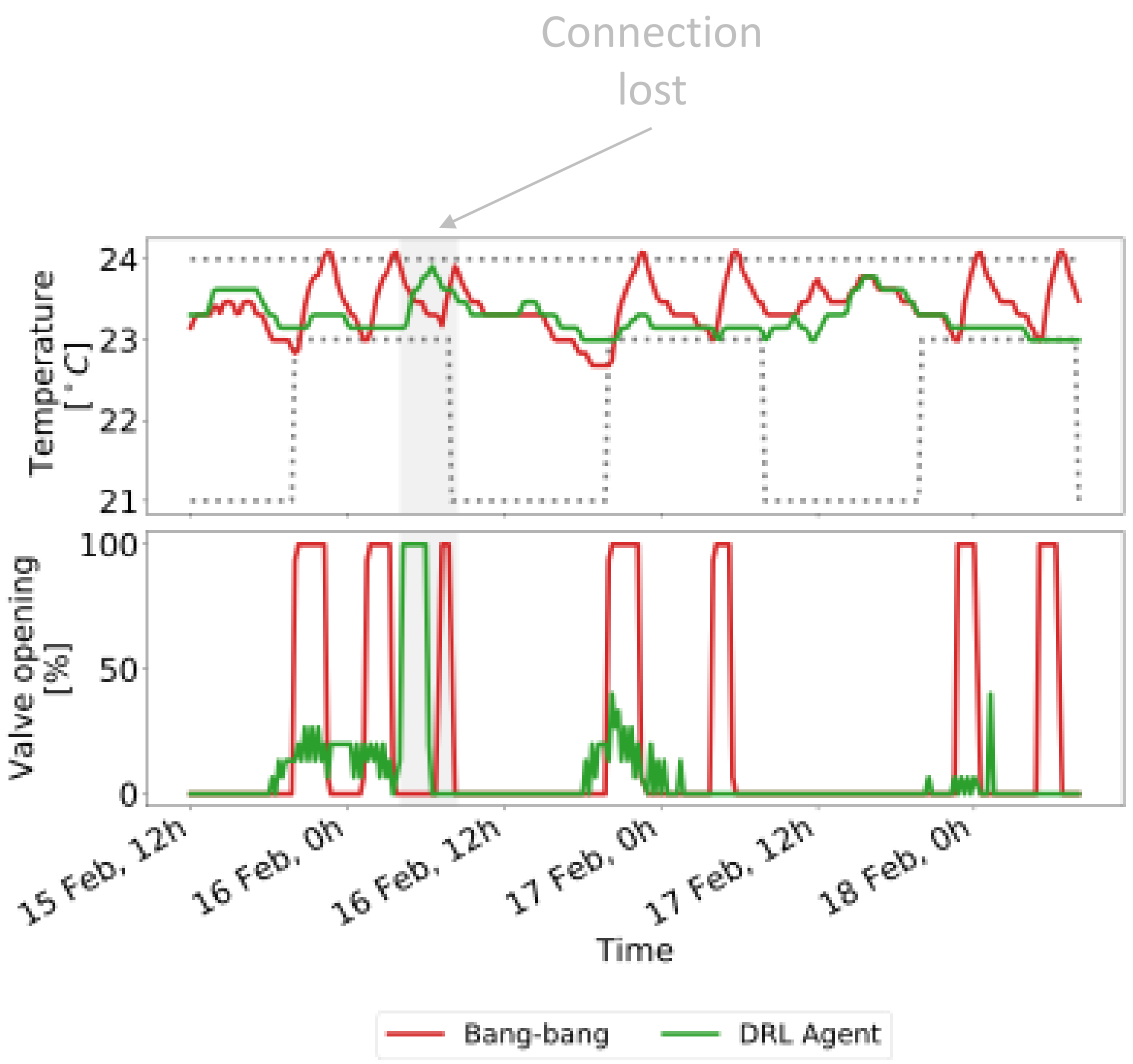
Simulation

- DRL agents **converge to behaviors similar to the optimal trajectories** (computed *a posteriori*)
  - Confirmed by **statistical analysis** over 2000 3-day long sequences
  - And for **different reward functions** and **random seeds**



Experiment

- Only short period because of connection issues
- But the agent shows **similar behavior as in simulation**
  - Preheating
  - Close to lower bound



• Introduction

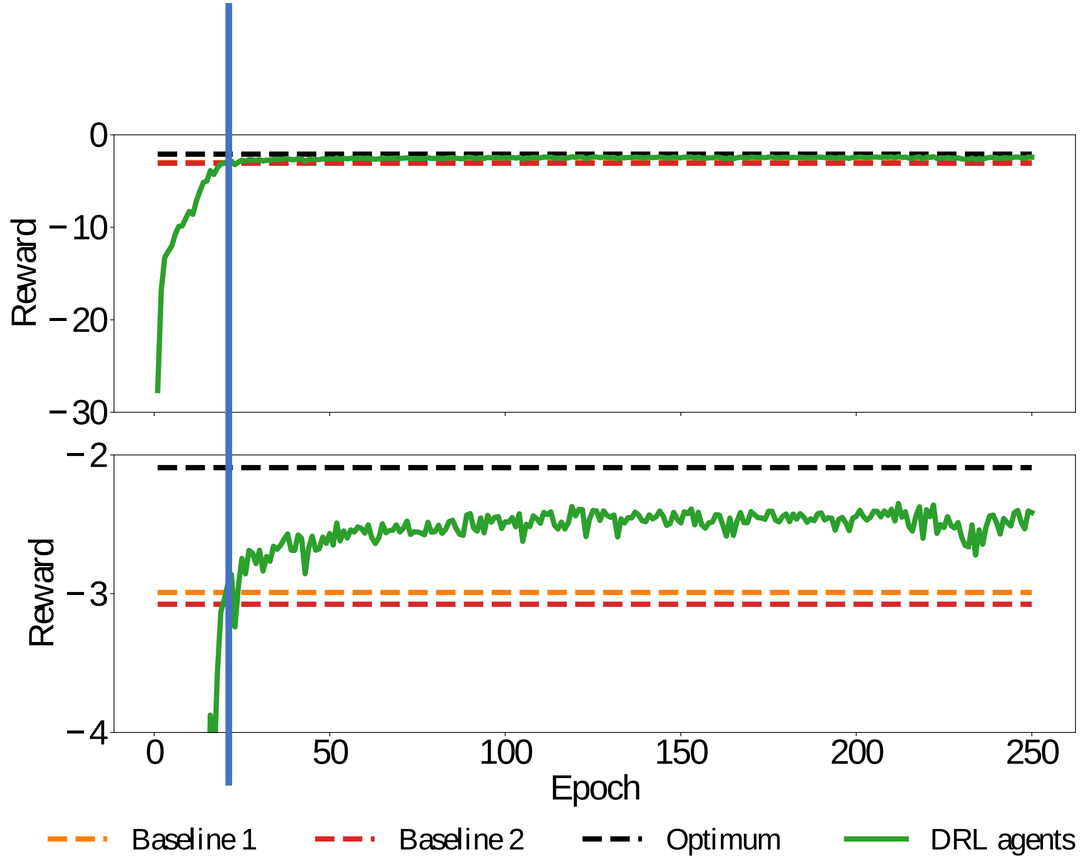
An issue of convergence speed

- **3 years** worth of data (randomly sampled days) to obtain the performance of classical rule-based controllers
- **15 years (500k points)** to “converge”
- (D)RL is usually *(very, very, very)* **data-inefficient**

High computational burden

Limited deployment on physical systems

100k steps
Roughly 3 years worh of data

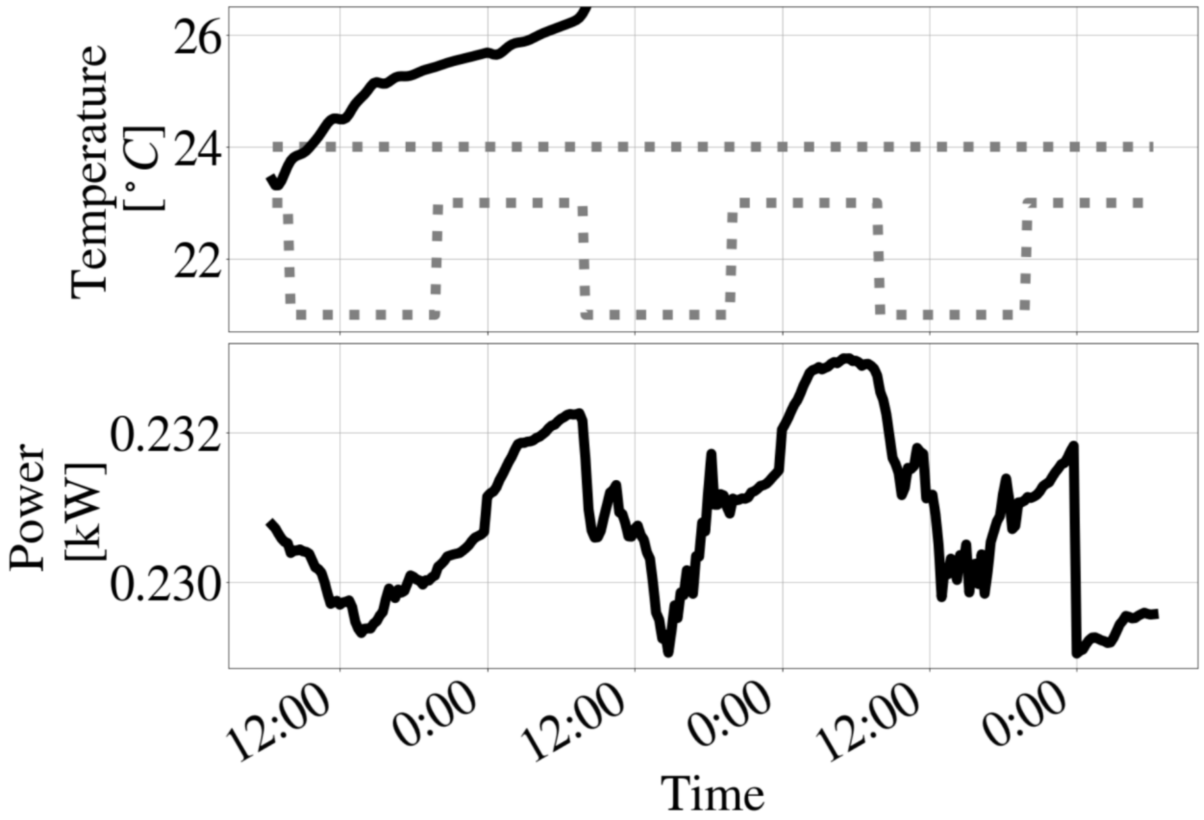


- Introduction

Why?

To get there…

… we need this!



Exploration is *necessary*

… but often *stupid*!

Postulate

*Prior expert knowledge often provides **intuition** about what optimal policies should do*

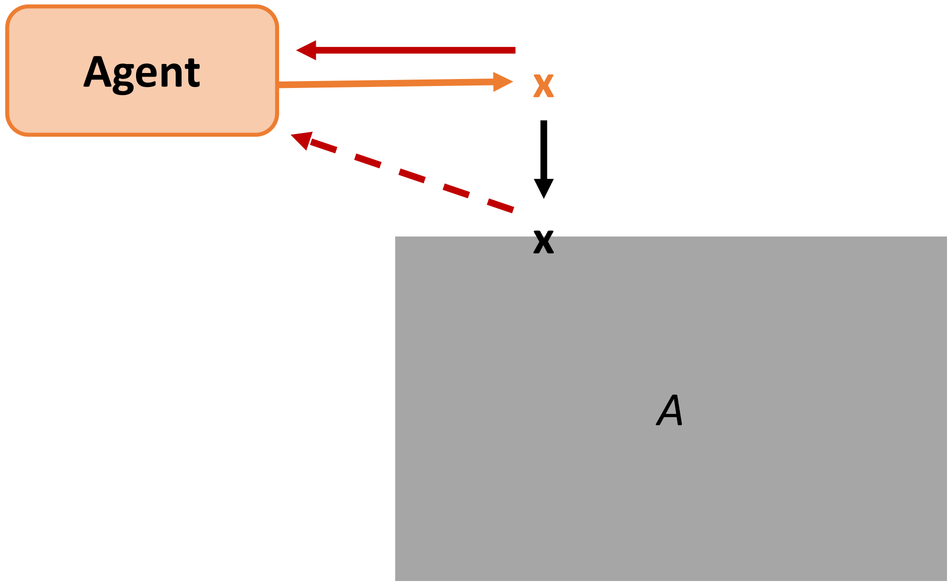
*We can accordingly **constrain the agents** to regions of the state space that are deemed interesting*

• Methods

Proposed approach to *accelerate learning*

How can we constrain the actions of agents to explore interesting states?

- Design **handcrafted simple rules** from expert knowledge to ensure meaningful actions
- **Saturate** the agents' actions according to these rules
- **Modify the gradient update** of the policy to let it learn from the corrections and steer it towards expected behaviors



Proposed approach to *accelerate learning*

How can we constrain the actions of agents to explore interesting states?

- **Correct the agents using safe action sets - but no need to always satisfy these artifical constraints!**

- Design **handcrafted simple rules** from expert knowledge to ensure meaningful actions
- **Saturate** the agents' actions according to these rules
- **Modify the gradient update** of the policy to let it learn from the corrections and steer it towards expected behaviors

$$a^{min}(s), a^{max}(s) = f(s)$$

$$a(s) = \text{clip}\left(\pi_\theta(s) + \nu, a^{min}(s), a^{max}(s)\right) \qquad \nu \sim \mathcal{N}(0, \sigma^2)$$

$$\hat{\nabla}_\theta^{EA} \pi_\theta = \nabla_\theta \left[\frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \pi_\theta(s)) - \frac{\lambda}{2} \left(\pi_\theta(s) - a(s)\right)^2 \right]$$

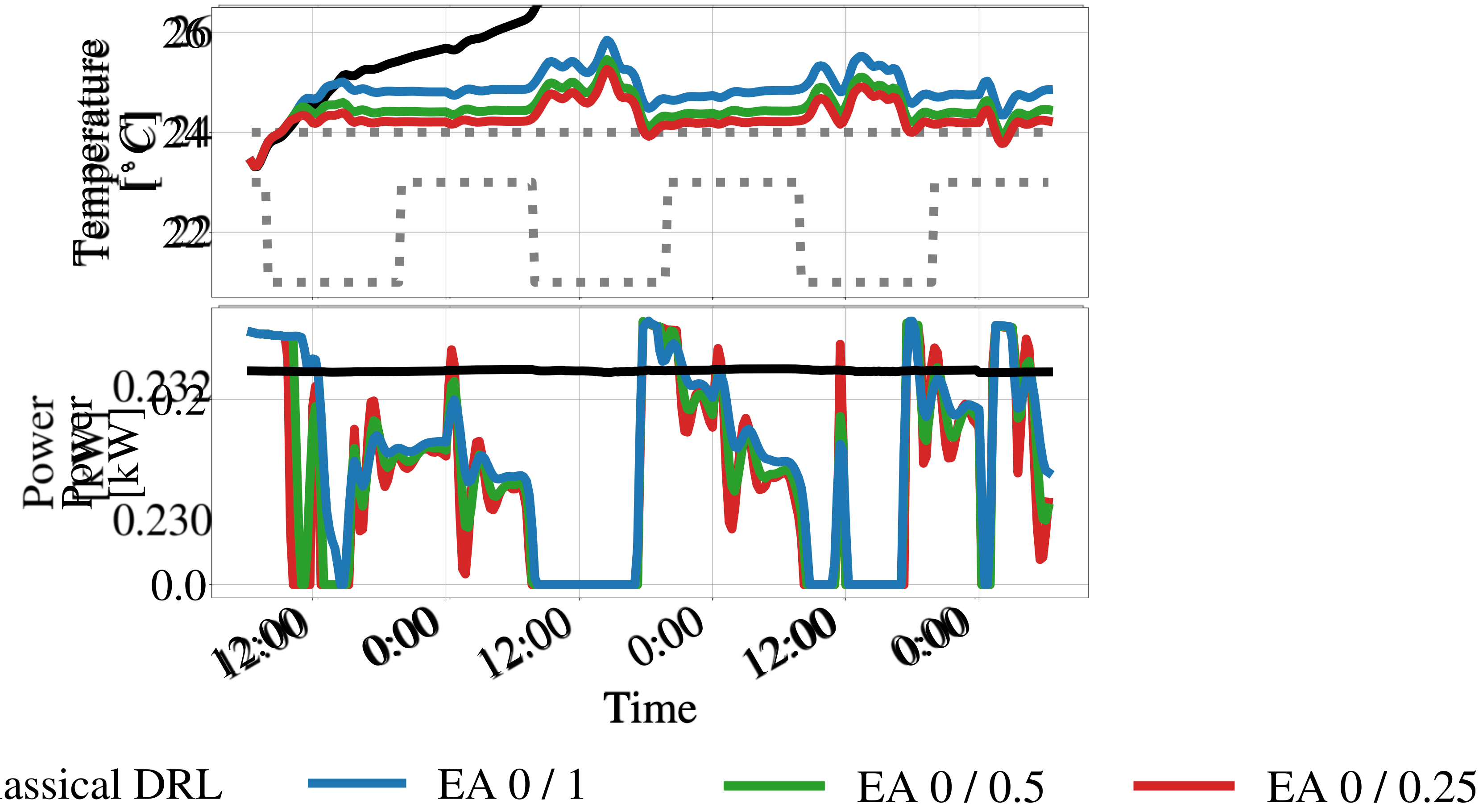
Each step is easy to implement and computationally inexpensive

• Introduction

What does the saturation do?

To get there…

… we need this!

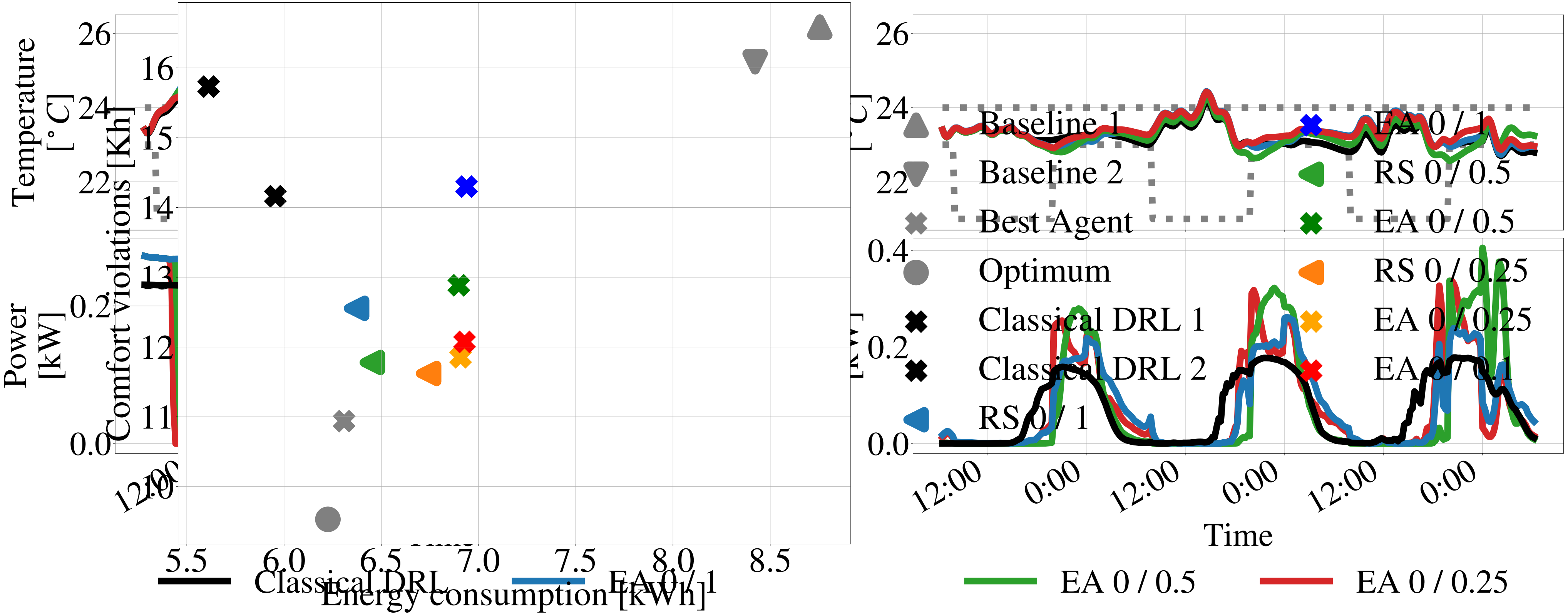


Meaningful exploration

- Results

Does it impact performance?

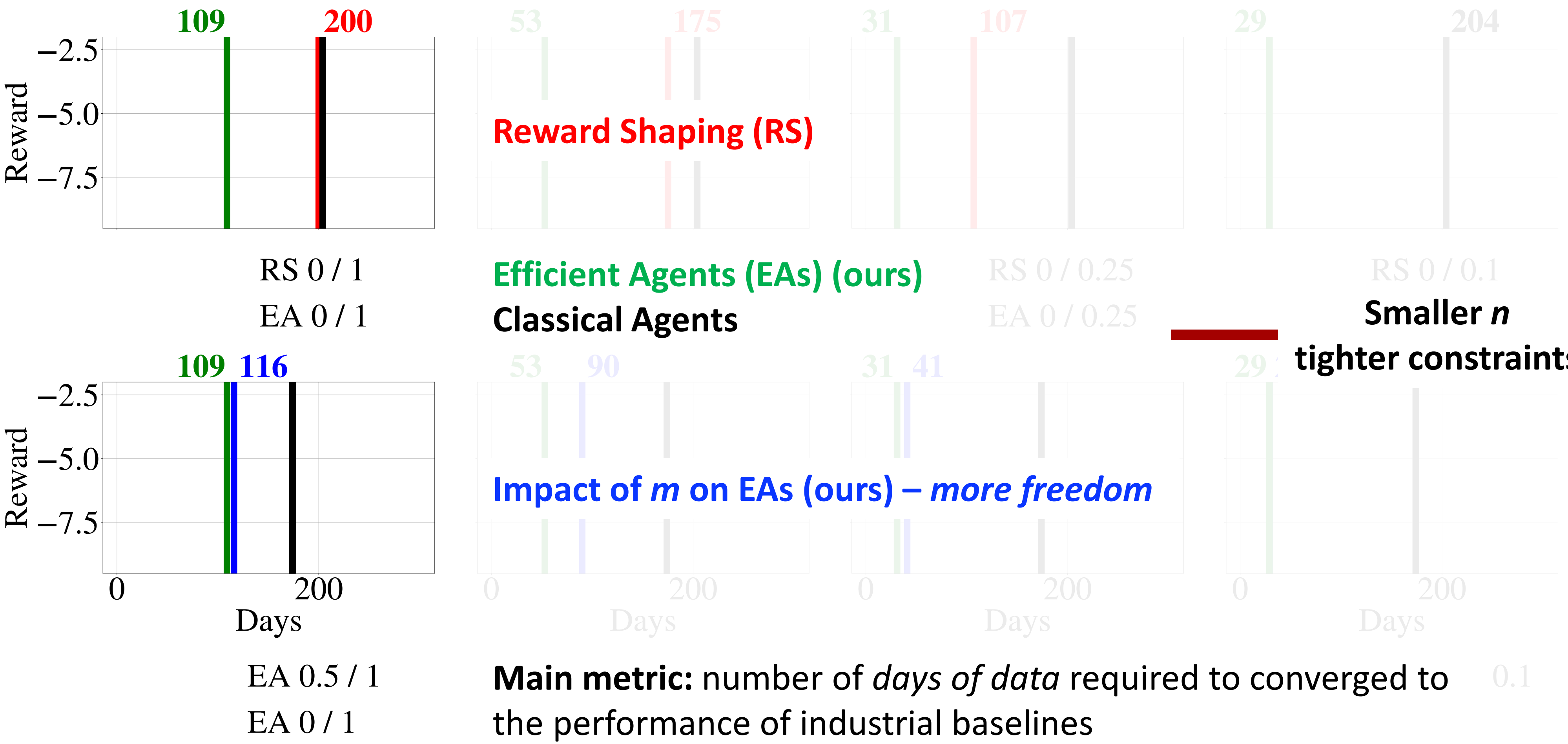
- The **learnt behaviors are similar!**



Good final performance

- Results

Does it accelerate learning?



Reward Shaping (RS)

Efficient Agents (EAs) (ours)

Classical Agents

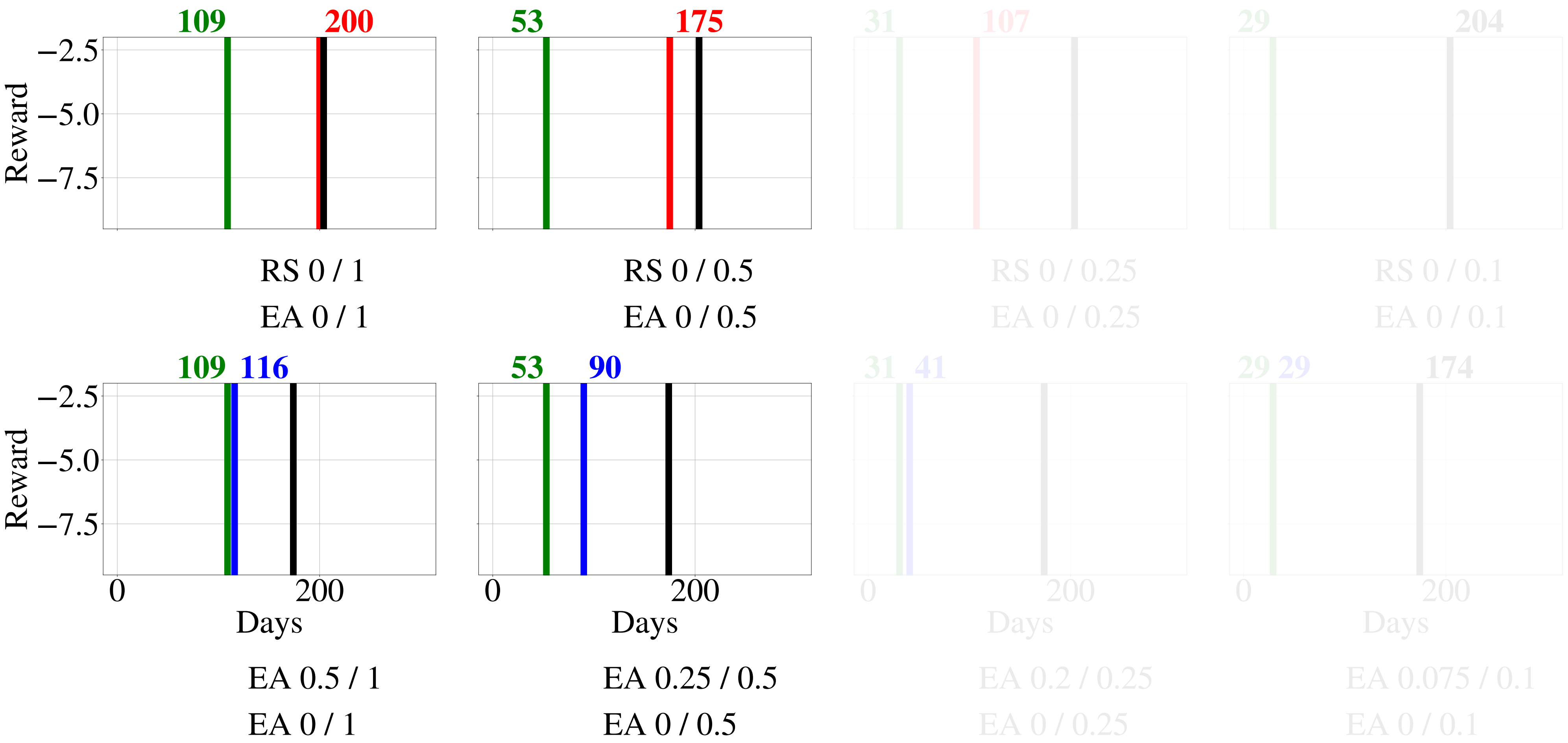
Smaller *n*
tighter constraints

Impact of *m* on EAs (ours) – *more freedom*

Main metric: number of *days of data* required to converged to the performance of industrial baselines

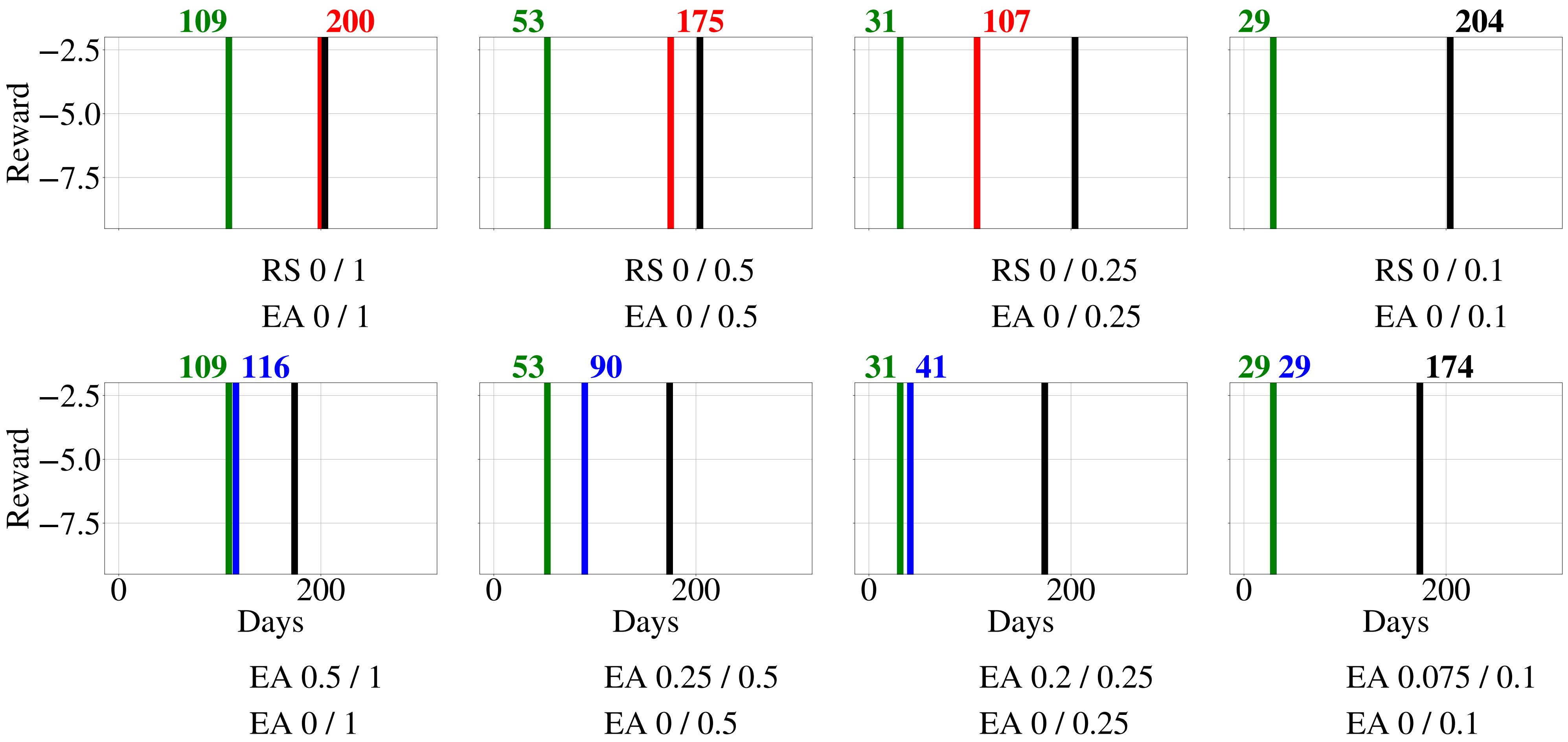
• Results

Does it accelerate learning?



- Results

Does it accelerate learning?



Objective attained! Up to 6-7x faster

- Wrap-up

Discussion & Conclusion

- **Black-box models can be misleading**, even if they fit the data well
  - PCNNs as a **potential solution - easy to scale**

→ ***Avoid tedious engineering***

- **Deep Reinforcement Learning** provides good results when coupled with PCNNs

→ ***Energy savings of 25-30%***
→ ***Comfort of the occupants improved***

- It is possible to design **simple rules based on prior knowledge** to help (D)RL agents converge faster
  - **Saturate** the actions of the agents accordingly
  - **Modify the gradient update** to let agents learn from it

→ ***Straightforward implementation***
→ ***Computationally inexpensive***
→ ***Up to 6-7 times faster***

→ Fully **black-box pipeline from data to control policies**

→ ***"Plug & Play" controllers***

Discussion & Conclusion

- **Neural Networks can be misleading**, even if they fit the data well
  - PCNNs as a **potential solution - easy to scale**

→ State-of-the-art accuracy
→ Physical consistency

- **Deep Reinforcement Learning** provides good results when coupled with PCNNs

→ Energy savings of 25-30%
→ Comfort of the occupants improved

- It is possible to design **simple rules based on prior knowledge** to help (D)RL agents converge faster
  - **Saturate** the actions of the agents accordingly
  - **Modify the gradient update** to let agents learn from it

→ Straightforward implementation
→ Computationally inexpensive
→ Up to 6-7 times faster

→ Fully **black-box pipeline from data to control policies**

→ "Plug & Play" controllers

Main references

PCNNs

- **Towards Scalable Physically Consistent Neural Networks: an Application to Data-driven Multi-zone Thermal Building Models.**
  L. Di Natale, B. Svetozarevic, P. Heer, and C.N. Jones (2022).
  Manuscript submitted to Applied Energy. https://arxiv.org/abs/2212.12380.

- **Physically consistent neural network building models: theory and analysis.**
  L. Di Natale, B. Svetozarevic, P. Heer, and C.N. Jones (2022).
  Applied Energy, 119806. https://doi.org/10.1016/j.apenergy.2022.119806.

DRL

- **Efficient Reinforcement Learning (ERL): Targeted Exploration Through Action Saturation.**
  L. Di Natale, B. Svetozarevic, P. Heer, and C.N. Jones (2022).
  Manuscript submitted to L4DC 2023. https://arxiv.org/abs/2211.16691.

- **Near-optimal Deep Reinforcement Learning Policies from Data for Zone Temperature Control.**
  L. Di Natale, B. Svetozarevic, P. Heer, and C.N. Jones (2022).
  2022 IEEE 17th International Conference on Control & Automation (ICCA), 698-703.
  https://doi.org/10.1109/ICCA54724.2022.9831914.