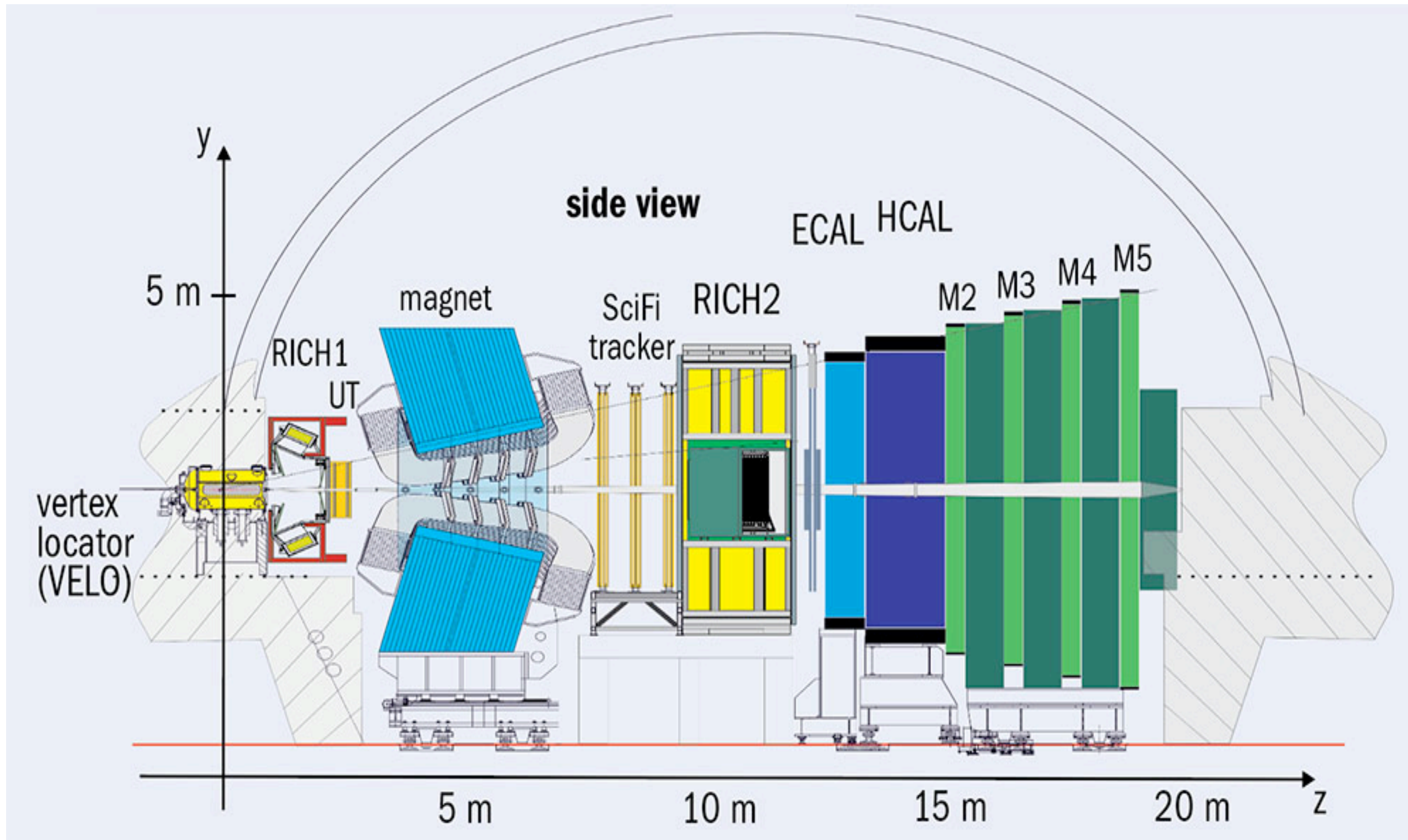




ML in LHCb

Prospecting for New Physics through Flavor, Dark Matter, and Machine Learning

Intro



- ML online
 - Trigger
 - ML for HLT
- ML offline
 - Robust PID
 - Flavor Tagging
 - Fast Simulation

The Trigger at LHCb

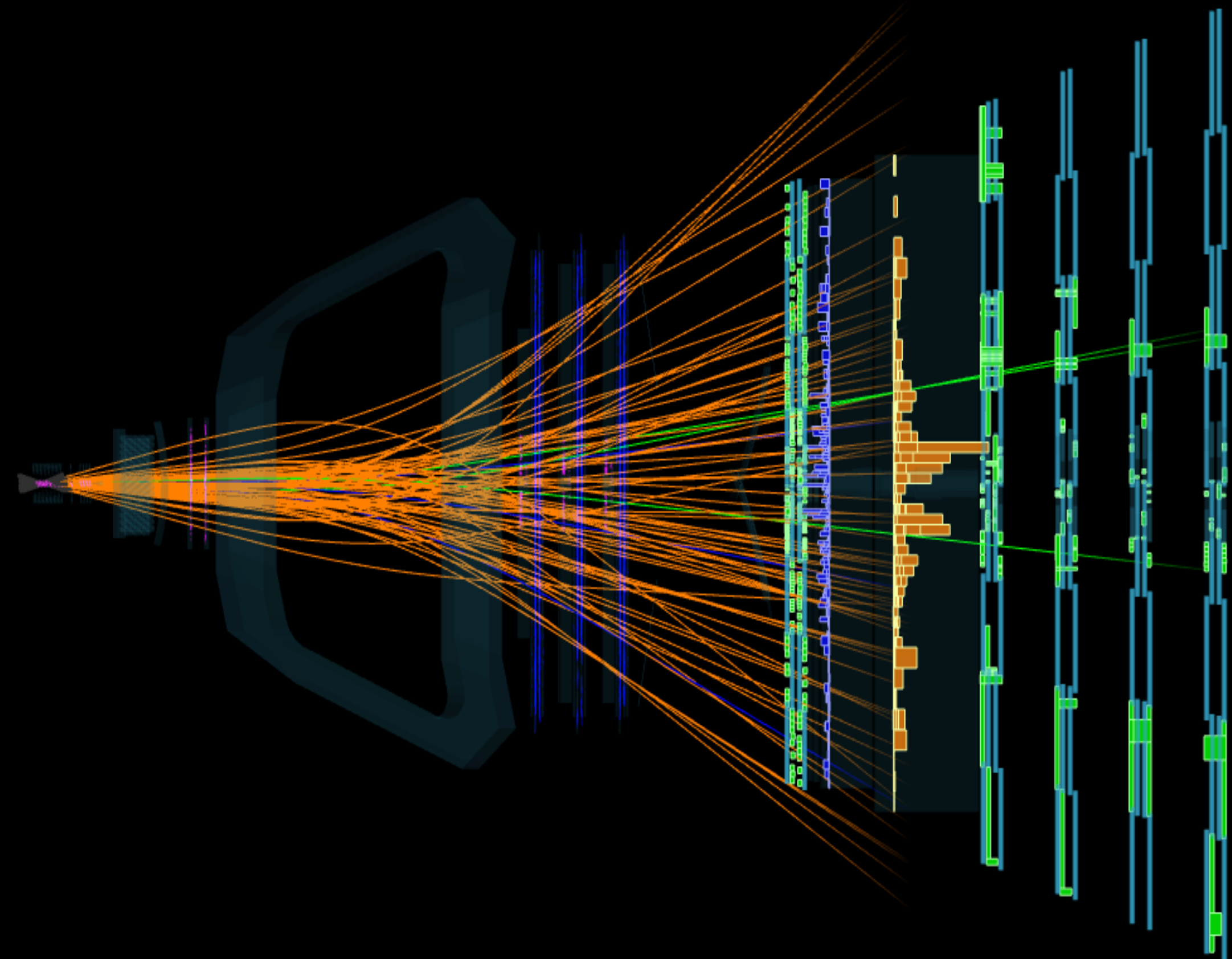
LHCb Raw data
15000 PB/year

LHCb storage capacity
30 PB/year



Select **only interesting** events:
Hybrid approach of expert systems and ML

Real time data reduction: 5 TB/s \rightarrow 10 GB/s



Event Selection -- data reduction

Trigger: mostly an expert system

Many subsystems look for particular decays

→ Strong reduction and purity ✓

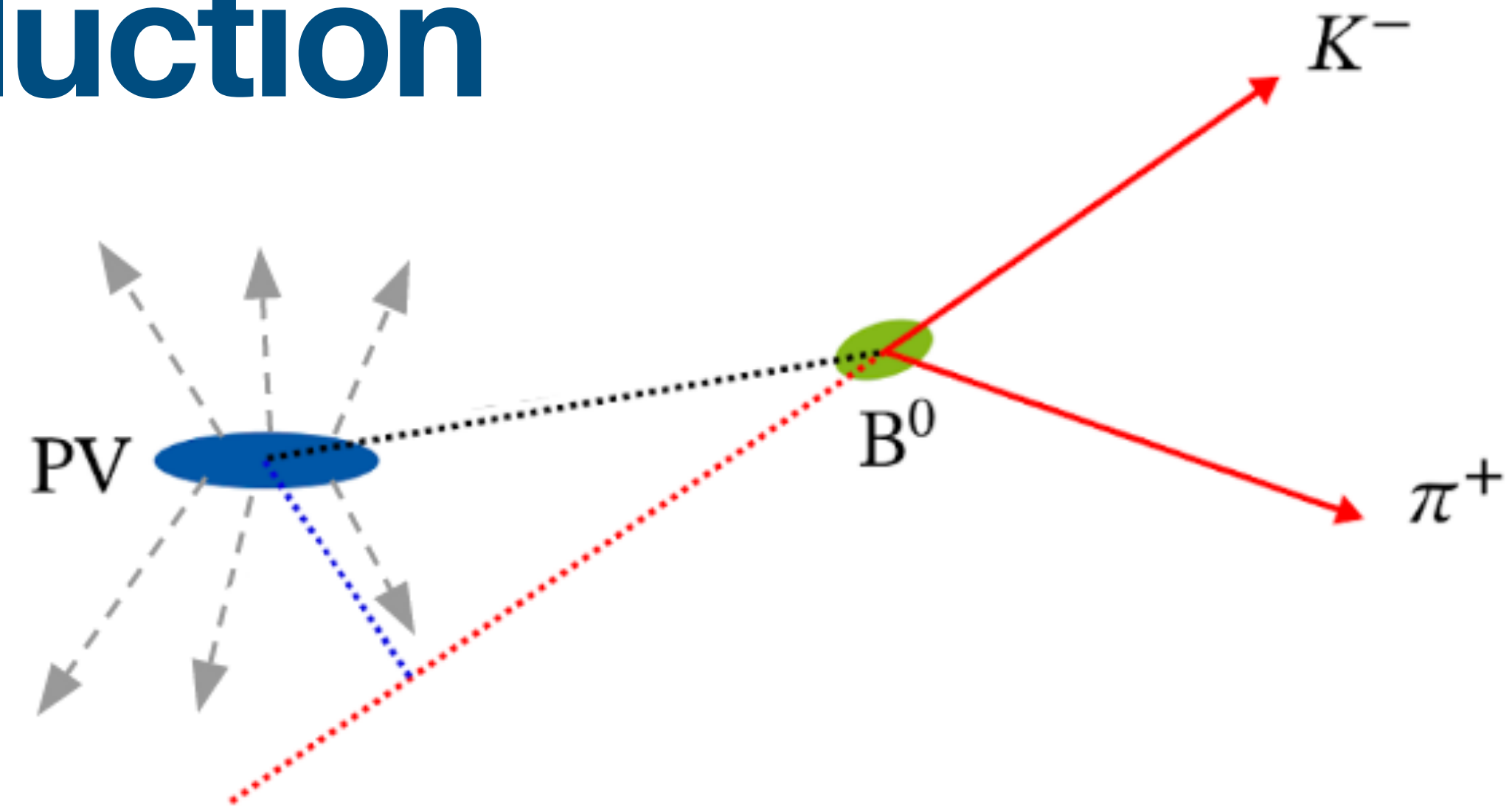
Some look for general signatures, weaker selection

→ Achieve good purity with ML classifiers

Need guarantees to employ these! **No room for error**

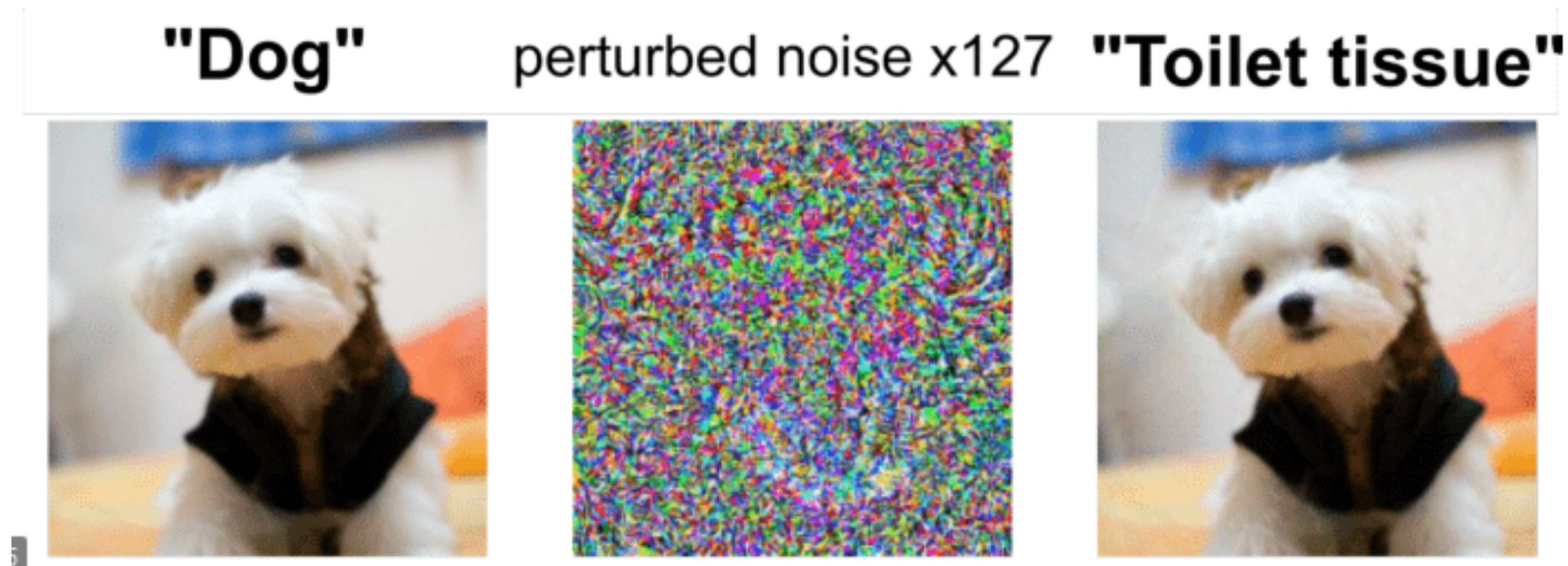
Guarantees needed:

1. Robustness w.r.t small changes
2. Monotonicity in certain features for "out of distribution" (OOD) guarantees



(Adversarial) Robustness

many SOTA ML models are proven to be highly unstable



$$\text{Robustness} := f(x + \epsilon) = f(x) + O(\epsilon)$$

I want deterministic robustness, i.e. provably robust networks!

Deterministic Robustness

NeurIPS ML4PS 21
arXiv:2112.00038

WLOG: Binary classifier: $F : \mathbb{R}^n \rightarrow \mathbb{R}$

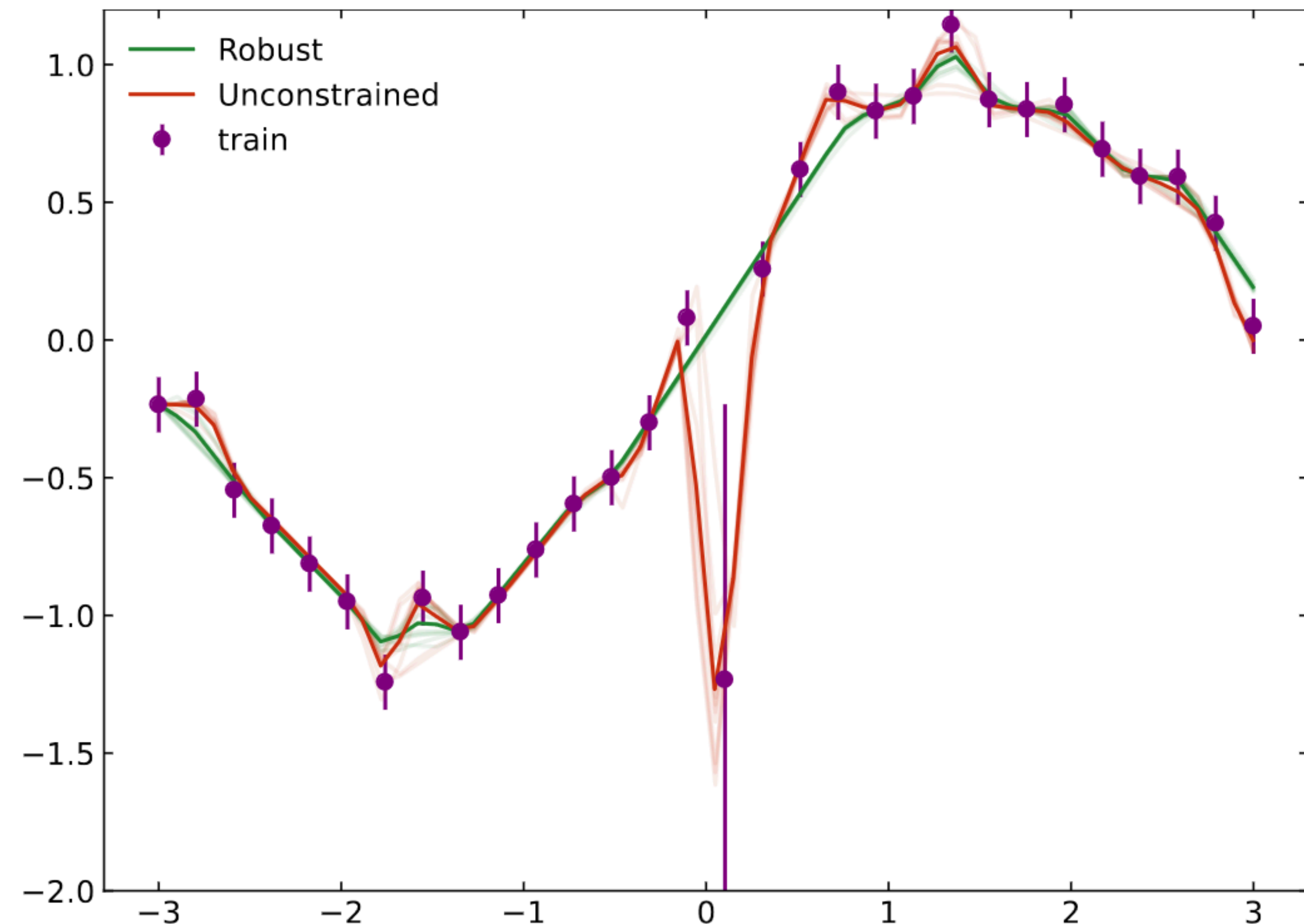
Constrain gradient wrt inputs, i.e. make it

Lipschitz- L $\|\nabla F\| \leq L$

A perturbation ϵ to an input x needs certain magnitude to flip the sign:

$$\text{sign } F(x + \epsilon) = -\text{sign } F(x)$$

$$\Rightarrow \|\epsilon\| > \frac{|F(x)|}{L}$$



Lipschitz Networks

NeurIPS ML4PS 21
arXiv:2112.00038

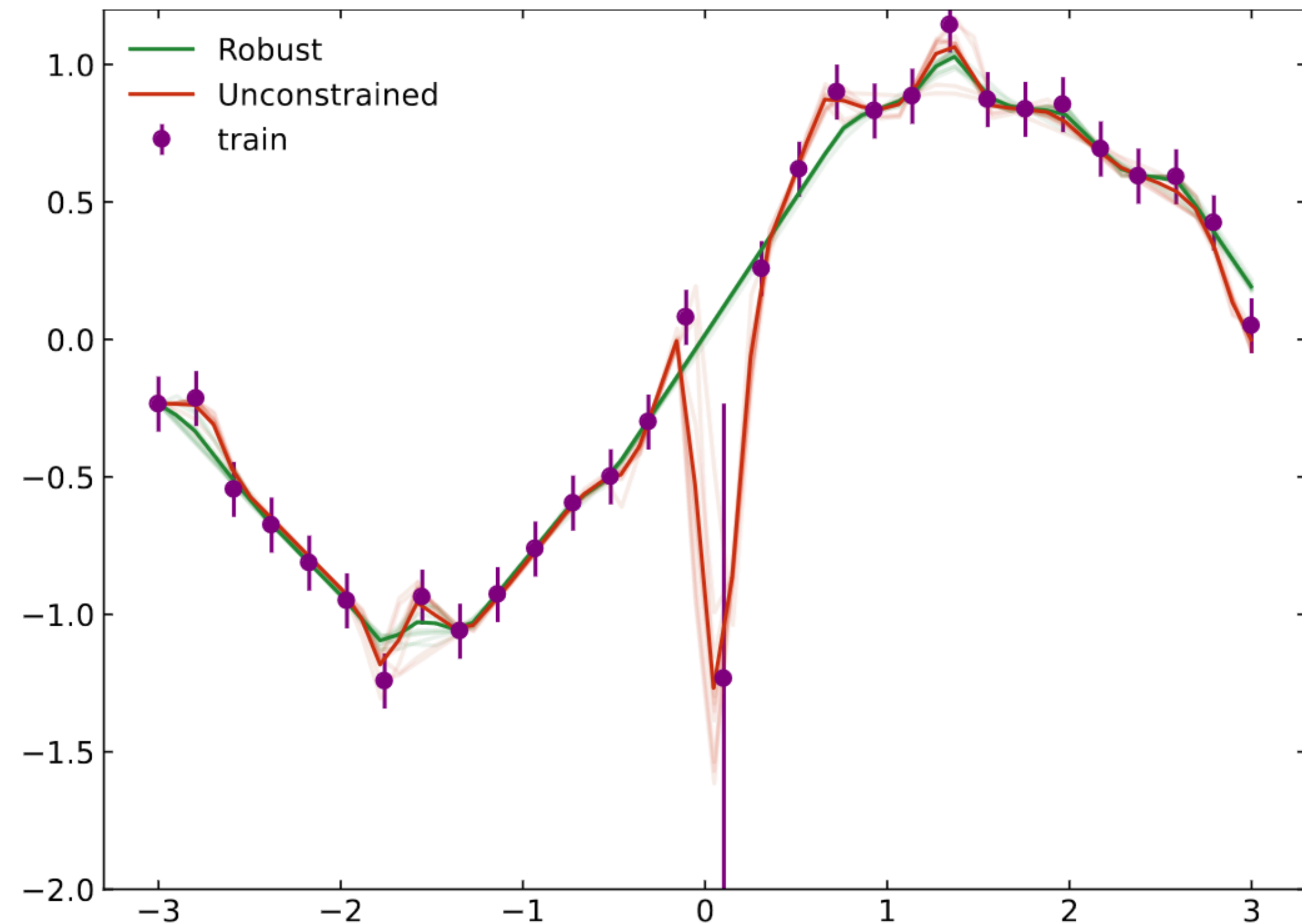
$\|\nabla F\| \leq L$ can be enforced by
constraining weights

In an MLP with Lipschitz-1 activations:

$$L \leq \prod \|W^i\|$$

(Toeplitz matrix for CNNs)

One possibility:
maintain $\|W^i\| \leq \sqrt[d]{L}$ in every layer



Monotonic Networks

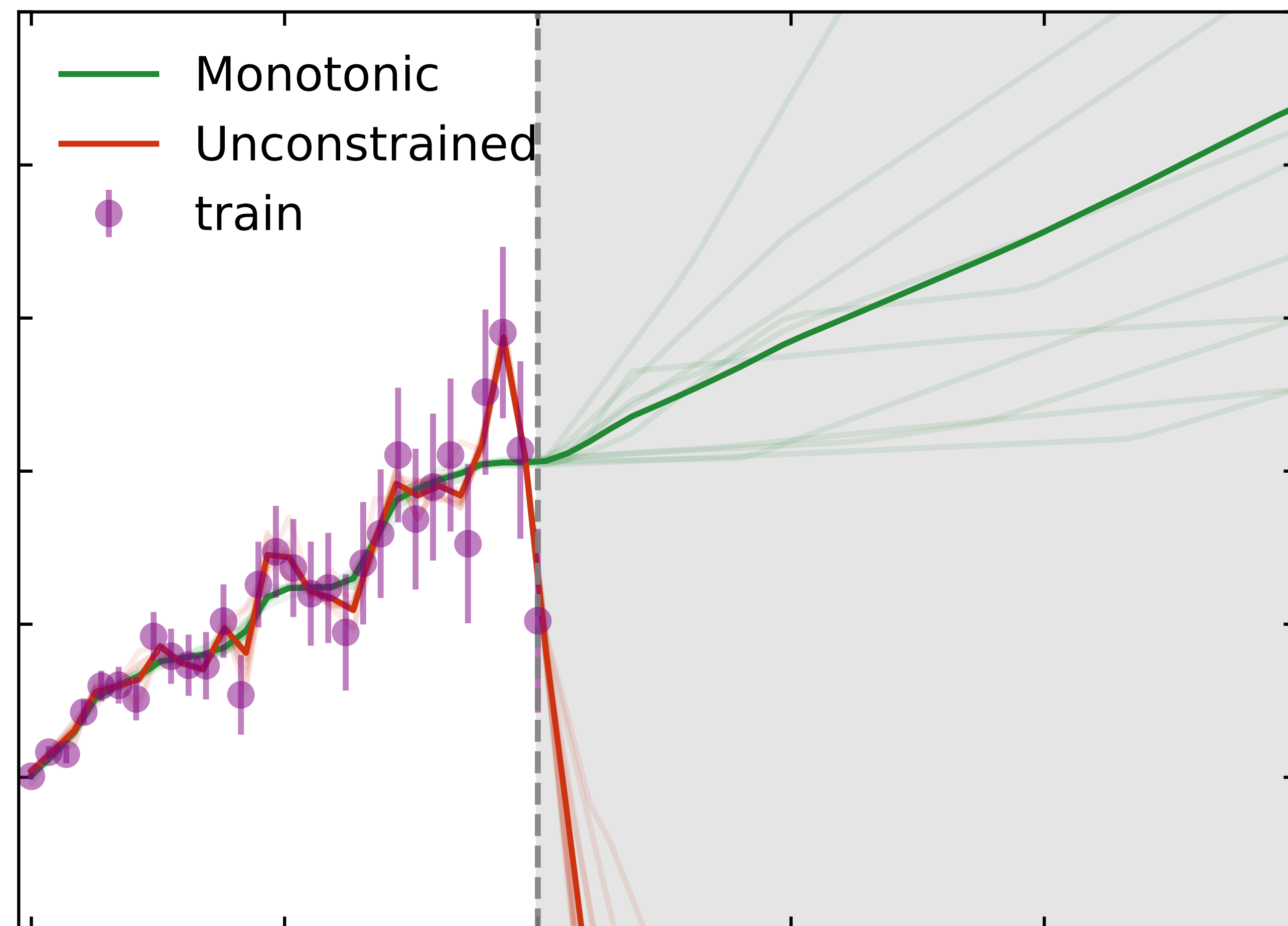
NeurIPS ML4PS 21
arXiv:2112.00038

Care about tails!

Guarantees about OOD
with monotonicity

Expressive monotonic networks
are not obvious.

Easier with robustness!



Monotonic Lipschitz Networks

NeurIPS ML4PS 21
arXiv:2112.00038

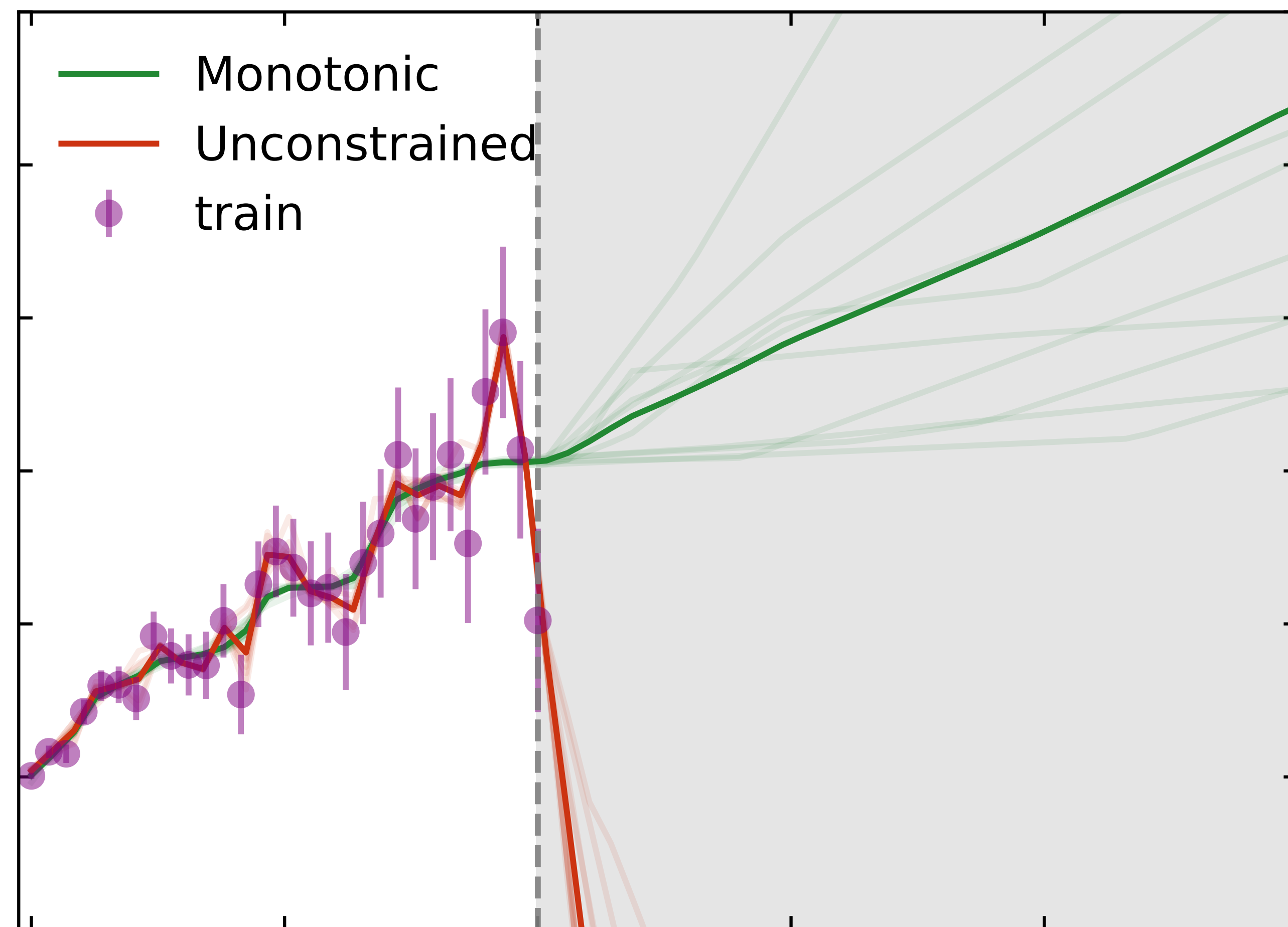
Combine Lipschitz networks
with monotonicity!

$$M(x) = F(x) + L \sum_i x_i$$

$$\frac{\partial M}{\partial x_i} = \frac{\partial F}{\partial x_i} + L \geq 0$$

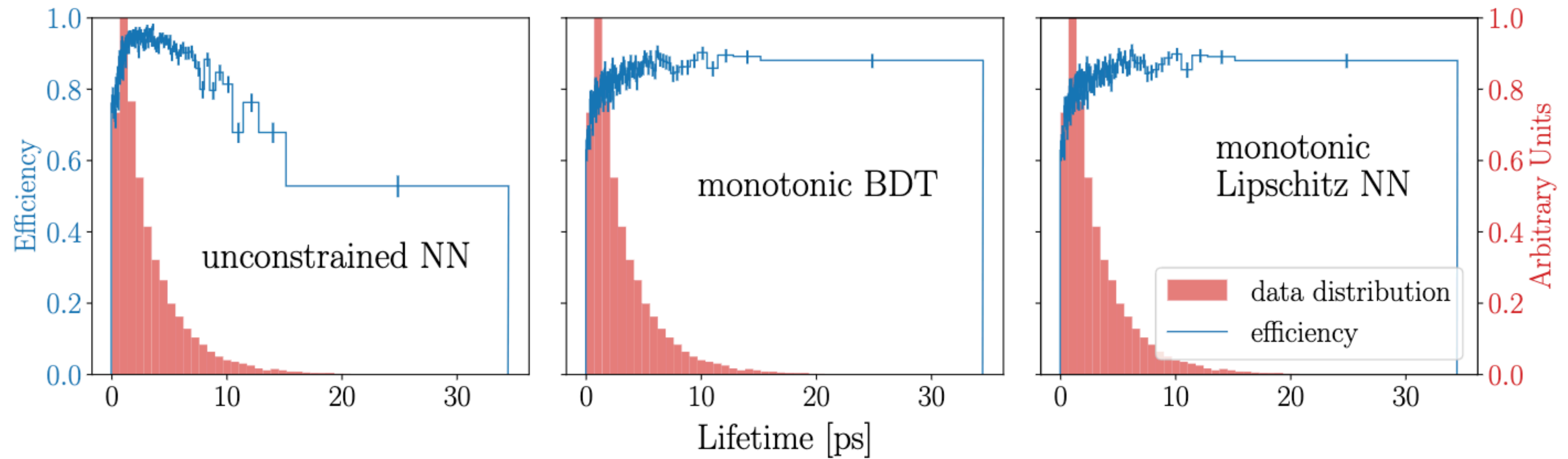


Lipschitz- L : $\|\nabla F\|_\infty \leq L$



Example

NeurIPS ML4PS 21
arXiv:2112.00038



Applications

HLT1 (on GPU)

1. fast inference with inclusive 1 & 2 trajectory beauty and charm selections
2. rejecting mis-reconstructed trajectories early in online reconstruction

HLT2

1. inclusive 2, 3 track beauty selection, Topological Trigger
2. in muon & and other more specialized signature selections

→ many instances of very small networks
(orthogonal to most current efforts in ML)

Bonus: Monotonicity as inductive bias appears often!
Applications in Medicine, Criminal Justice

ICLR 23
https://openreview.net/forum?id=w2P7fMy_RH

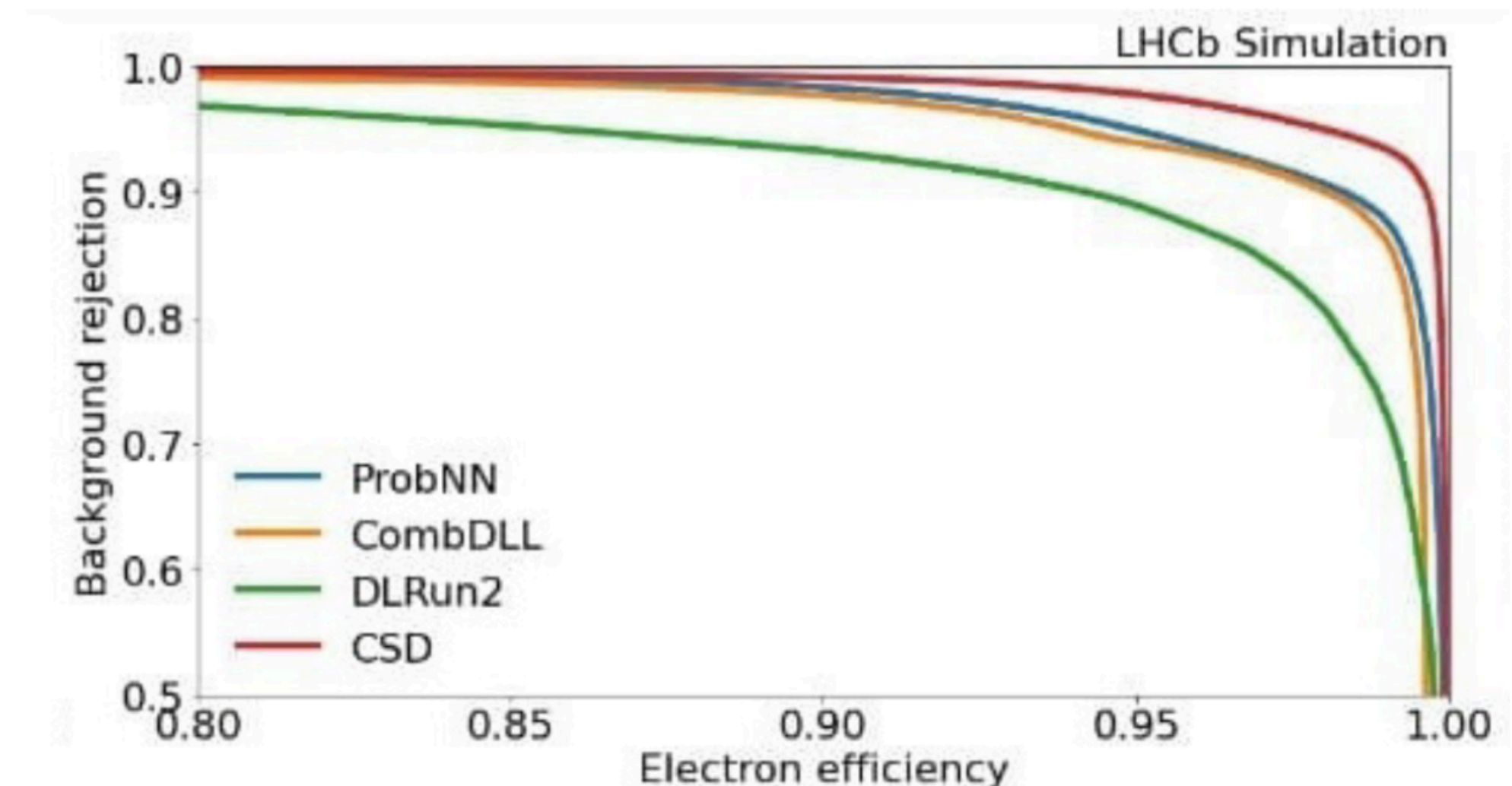
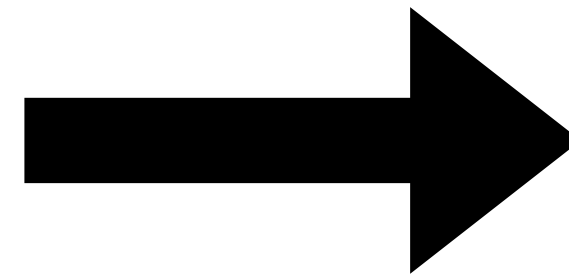
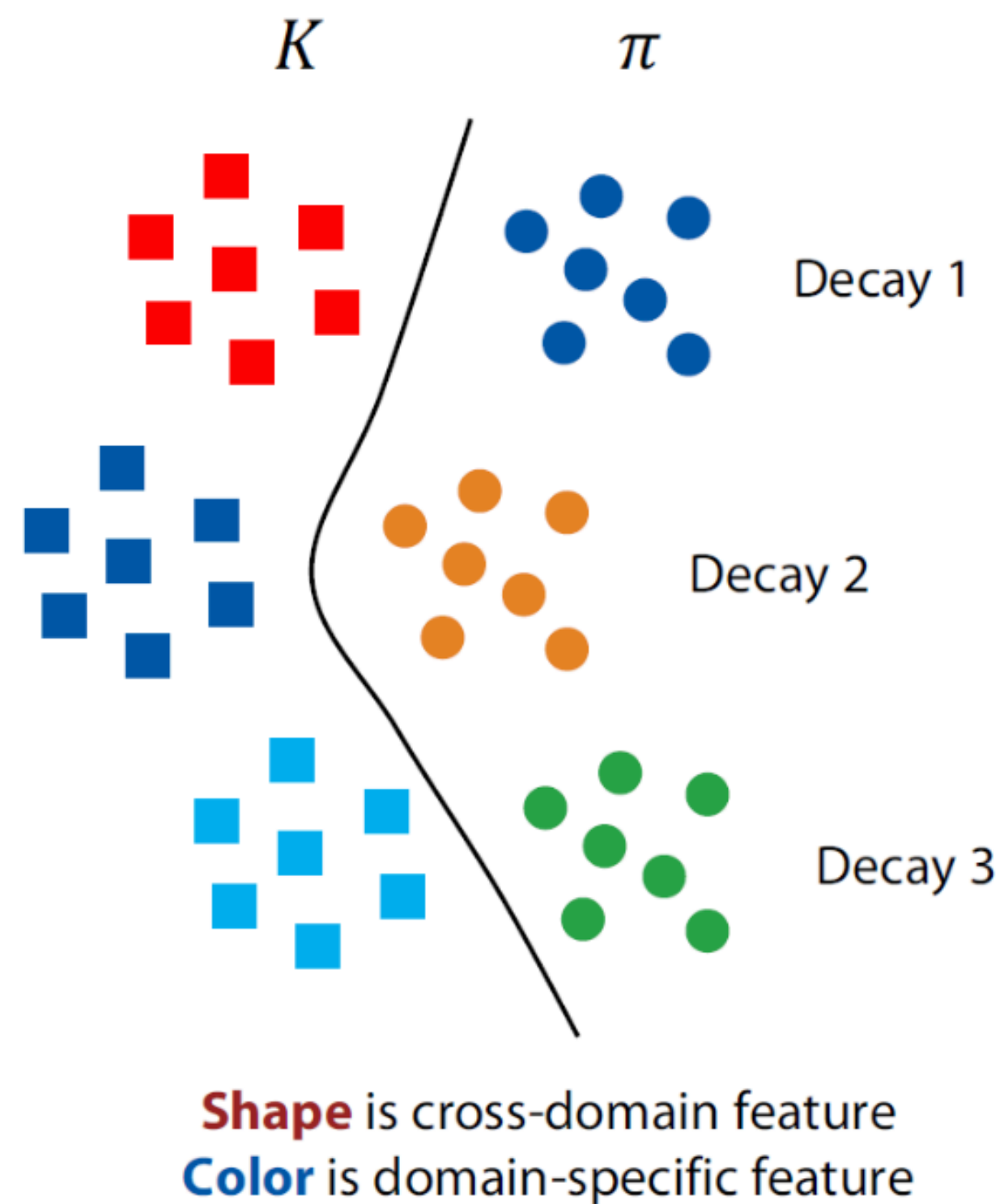
ML offline

1. Robust PID
2. Flavor Tagging
3. Fast Simulation
4. More

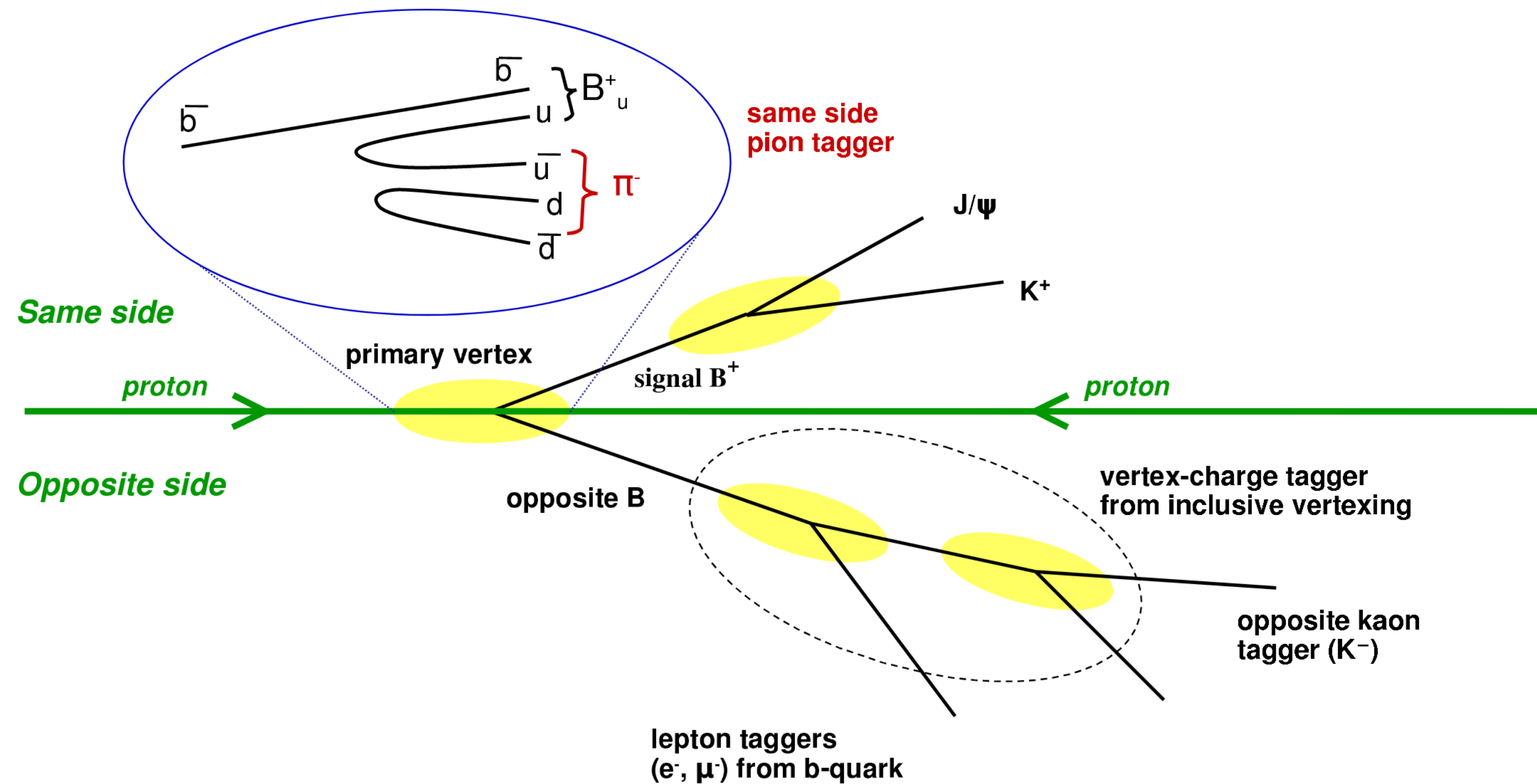
Robust Particle Identification

arXiv:2212.07274

Slightly different definition of robust: w.r.t. domain shifts via Common-Specific Decomposition (CSD)



Flavor Tagging



Input

The full event
[#tracks, #features]

Output

B_0 or \bar{B}_0

Classical taggers

Find particles that indicate
the B flavor

ML Taggers

Predict flavor directly

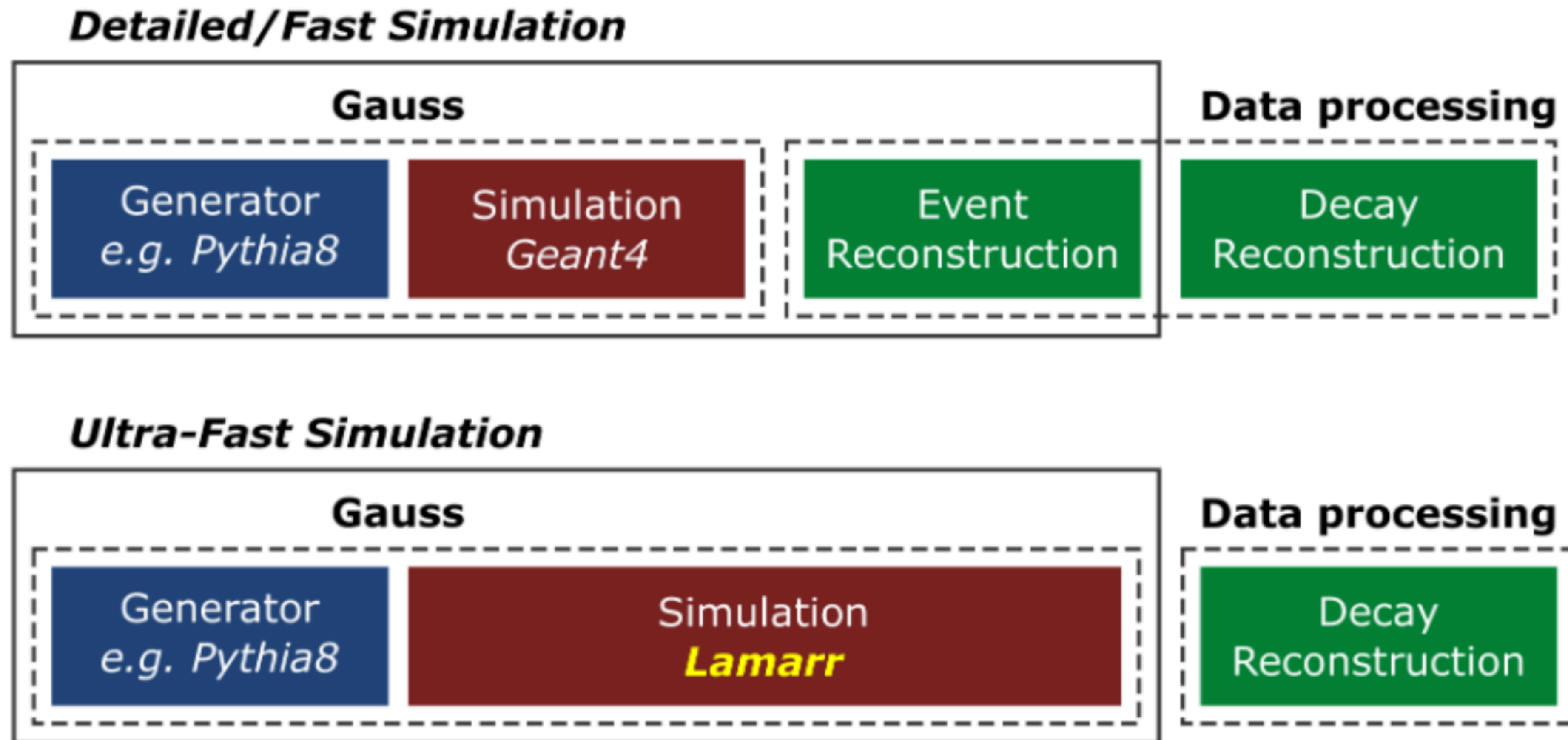
RNNs

DeepSets

Transformers

up to **50% more tagging
power** with ML taggers!

Fast Simulation - Parametrize the Sim Pipeline



Two Approaches

Efficiencies

Gradient Boosted Decision Trees (GBDTs) trained on simulated data with *Binary* or *Categorical Cross Entropy* to predict the fraction of “good*” candidates, *i.e.* the “efficiency” of a specific step as a function of generator-level quantities.

- GBDTs are robust and easy to train
- Almost no preprocessing is needed

* either “accepted”, “reconstructed”, “selected”... depending on the context

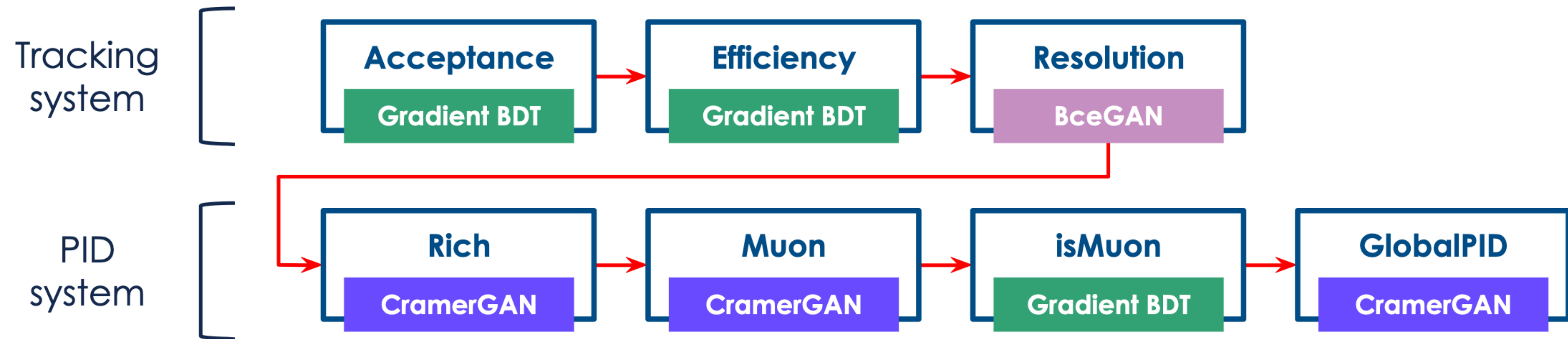
Reconstructed quantities

Conditional **Generative Adversarial Networks** trained on either simulated or calibration data.

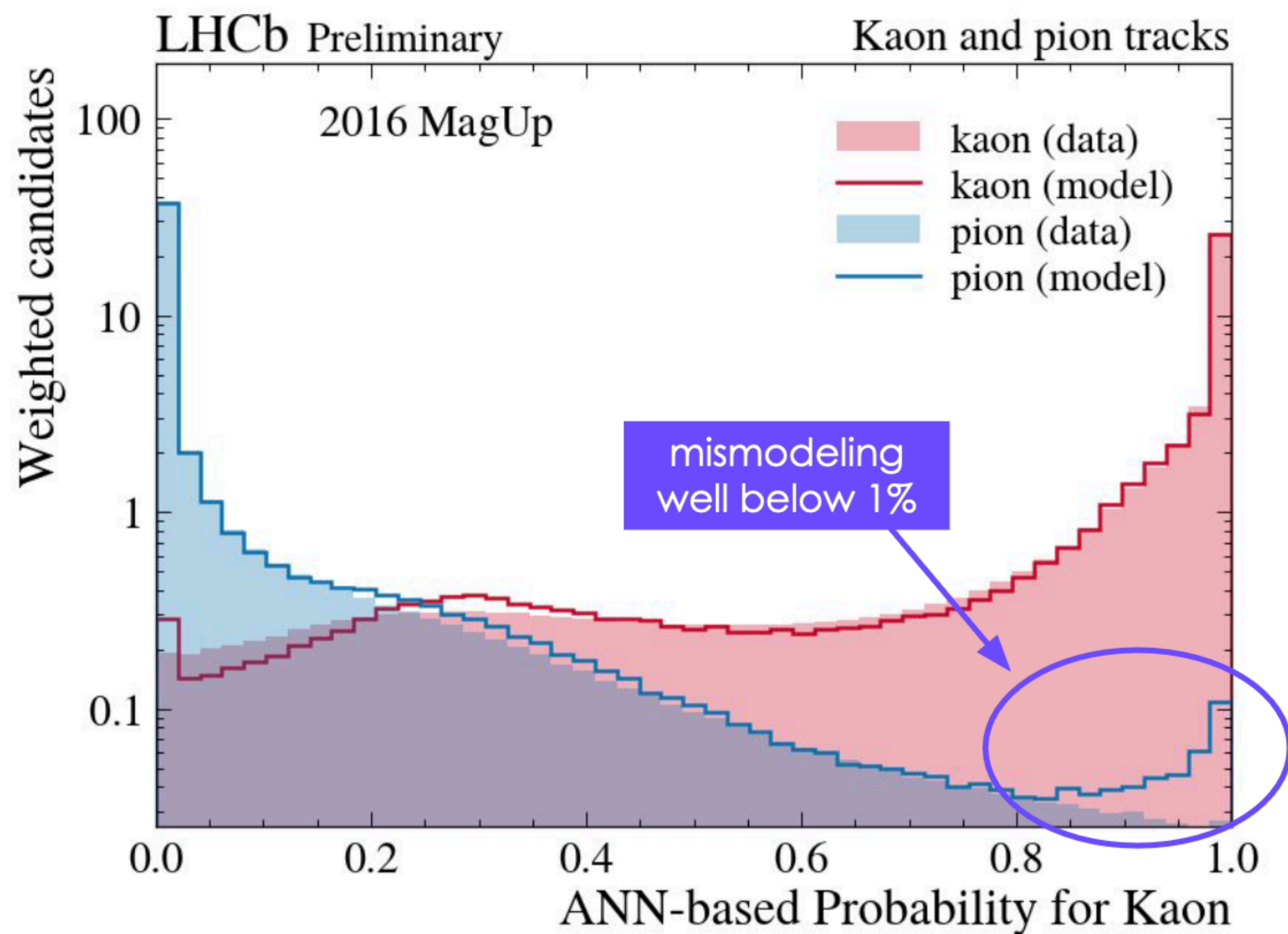
Various GAN flavours adopted for different parameterizations balancing between accuracy and robustness.

Training is performed on **opportunistic GPU resources** provided to the Collaboration.

Pipelines of Parametrizations



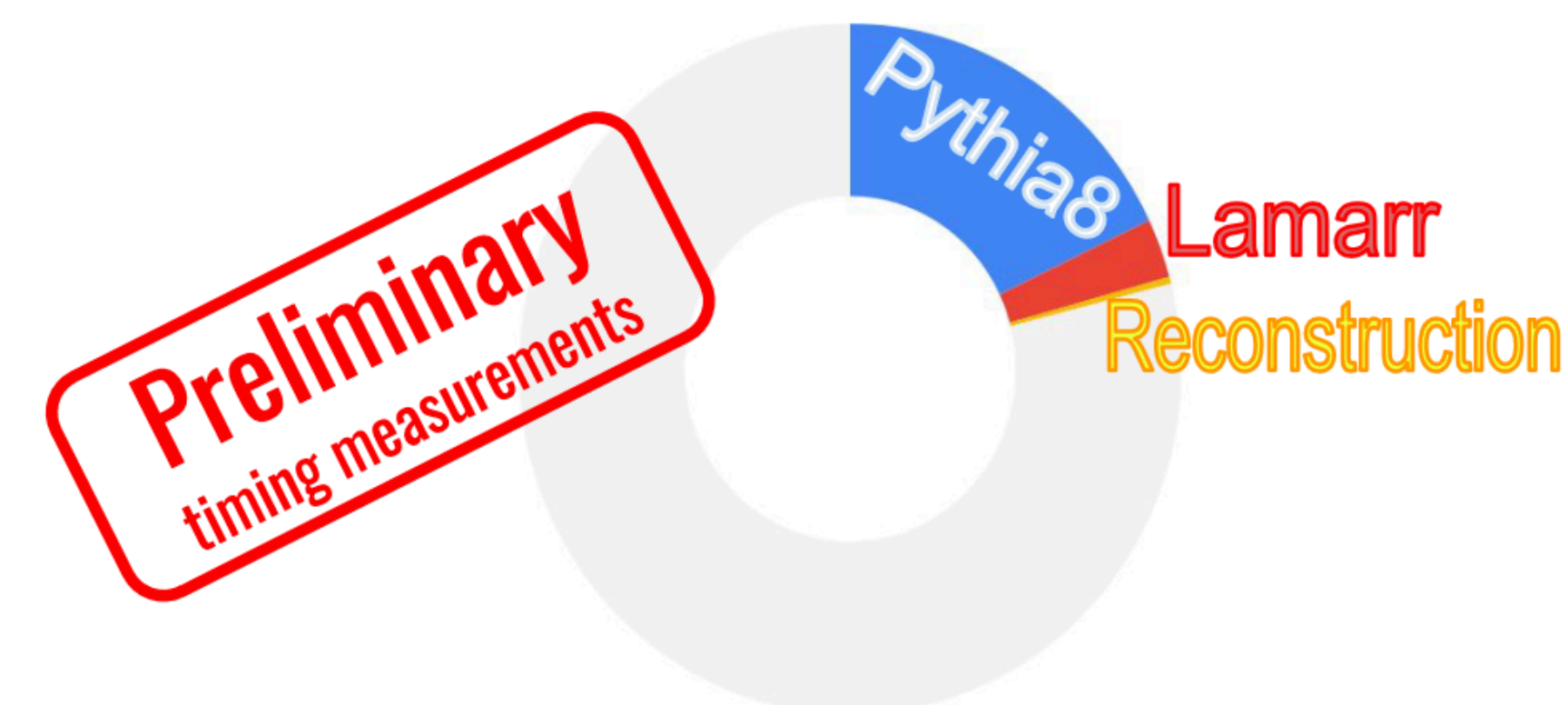
Pretty impressive!



Detailed simulation: Pythia8 + Geant4
 1M events @ 2.5 kHS06.s/event \approx 80 HS06.y



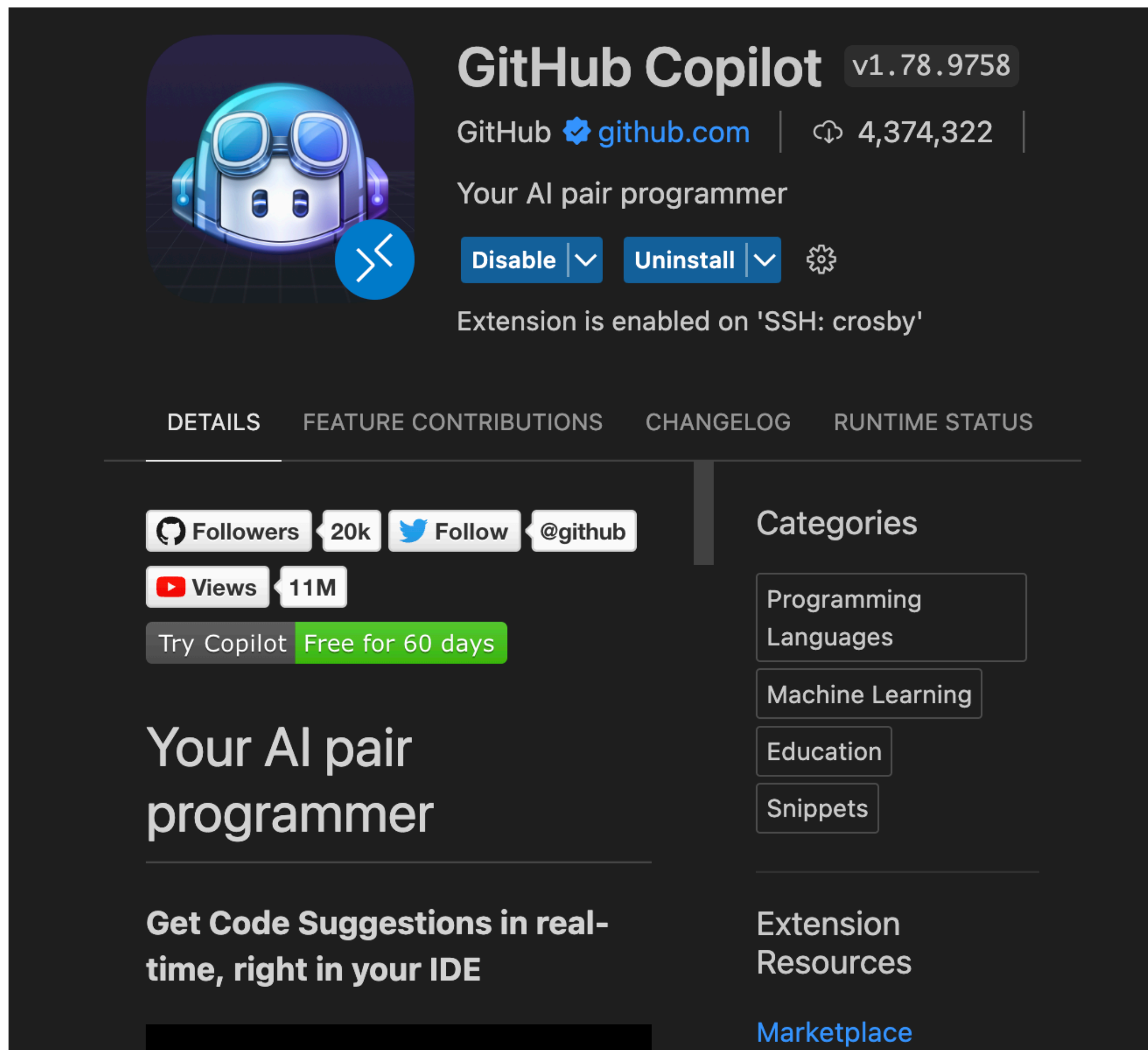
Ultra-fast simulation: Pythia8 + Lamarr
 1 M events @ 0.5 kHS06.s/event \approx 15 HS06.y



Etc Etc...

1. RoboShifter - Classify the quality of a run during data taking
doi:10.1088/1742-6596/898/9/092027
2. Particle ID with neural networks in the trigger
doi:10.1051/epjconf/201921406011
3. Particle ID with gaussian mixture models in the fixed target program (SMOG)
arxiv:2110.10259v2
4. Jet Tagging doi:10.1088/1748-0221/10/06/P06013
5. Decorrelation methods for ML in analyses arXiv:2010.09745, arXiv:1305.7248
6. And everything else that people come up with in analyses!

How we actually use ML

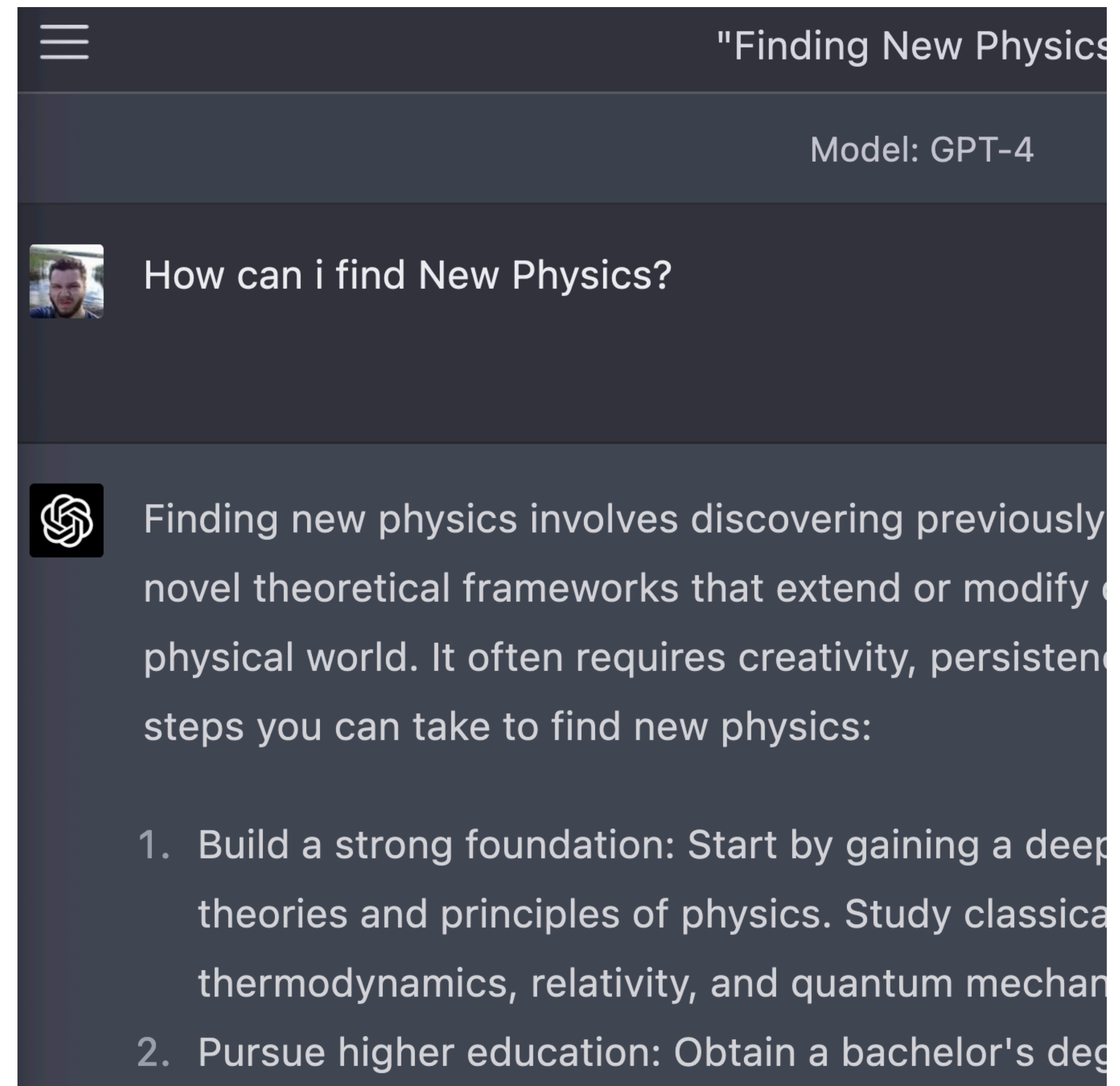


The screenshot shows the GitHub Copilot extension page. At the top, there's a header with the GitHub Copilot logo (a blue robot head) and the text "GitHub Copilot v1.78.9758". Below this, it says "GitHub github.com" and "4,374,322" downloads. The main text reads "Your AI pair programmer". There are buttons for "Disable" and "Uninstall", both with dropdown arrows, and a settings gear icon. Below these buttons, it says "Extension is enabled on 'SSH: crosby'".

Below the header, there are tabs for "DETAILS", "FEATURE CONTRIBUTIONS", "CHANGELOG", and "RUNTIME STATUS". Under the "DETAILS" tab, there are social media links: "Followers 20k", "Follow @github", and "Views 11M". There's also a "Try Copilot Free for 60 days" button.

On the right side, there's a "Categories" section with buttons for "Programming Languages", "Machine Learning", "Education", and "Snippets". At the bottom, there's an "Extension Resources" section with a link to the "Marketplace".

At the bottom left, there's a section titled "Your AI pair programmer" with the text "Get Code Suggestions in real-time, right in your IDE".



The screenshot shows a chat interface with a dark background. At the top, there's a header with a hamburger menu icon and the text "Finding New Physics". Below this, it says "Model: GPT-4".

The chat history shows a user message: "How can i find New Physics?". Below this, there's a response from GPT-4, which starts with the OpenAI logo and the text "Finding new physics involves discovering previously novel theoretical frameworks that extend or modify the physical world. It often requires creativity, persistence, and a deep understanding of the physical world. Here are some steps you can take to find new physics:".

The response is followed by a numbered list:

1. Build a strong foundation: Start by gaining a deep understanding of the theories and principles of physics. Study classical mechanics, thermodynamics, relativity, and quantum mechanics.
2. Pursue higher education: Obtain a bachelor's degree in physics or a related field.

Backup

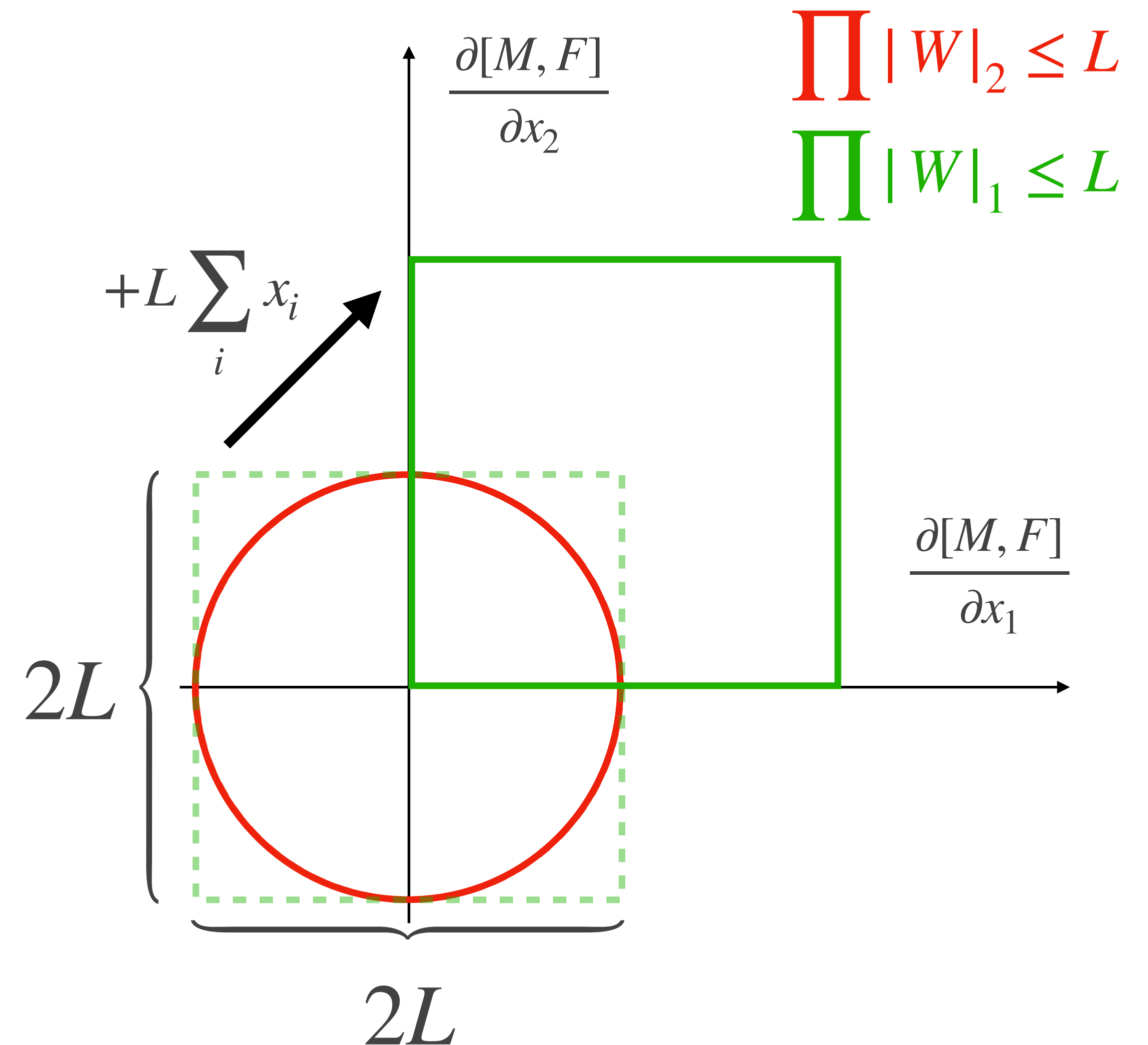
Monotonic Lipschitz Networks

$$M(x) = F(x) + L \sum_i x_i$$

$$\frac{\partial M}{\partial x_i} = \frac{\partial F}{\partial x_i} + L$$

$+L$ contribution in every direction x_i
 $\|\nabla F\| \leq L$ is not good enough

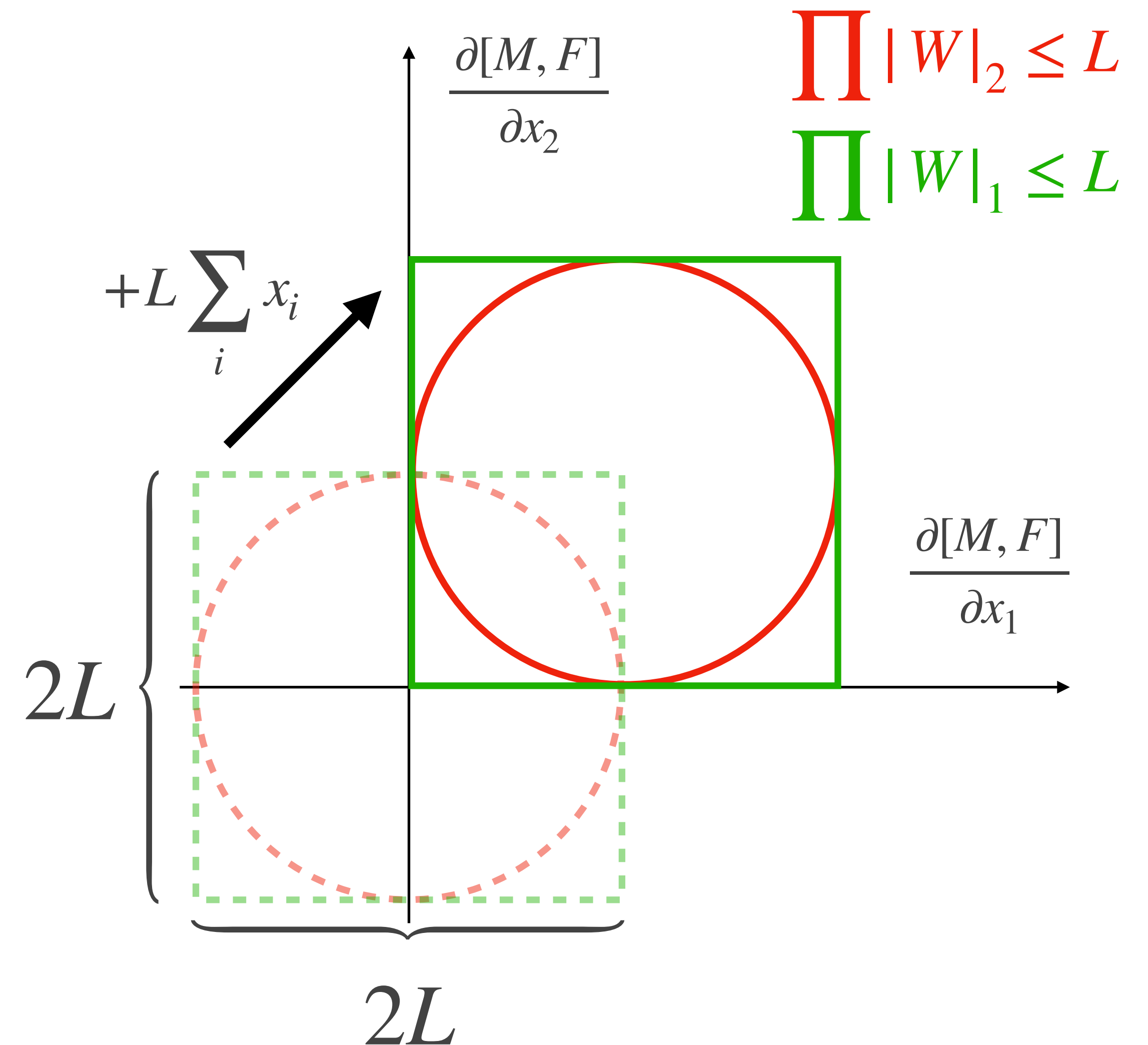
We want $\|\nabla F\|_\infty \leq L$!



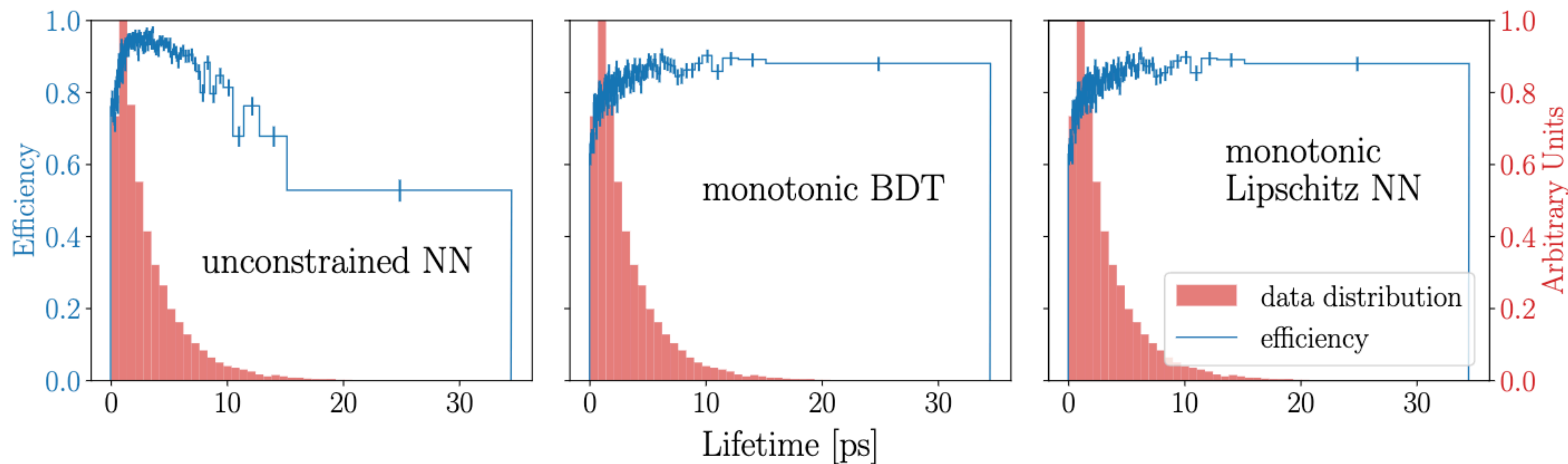
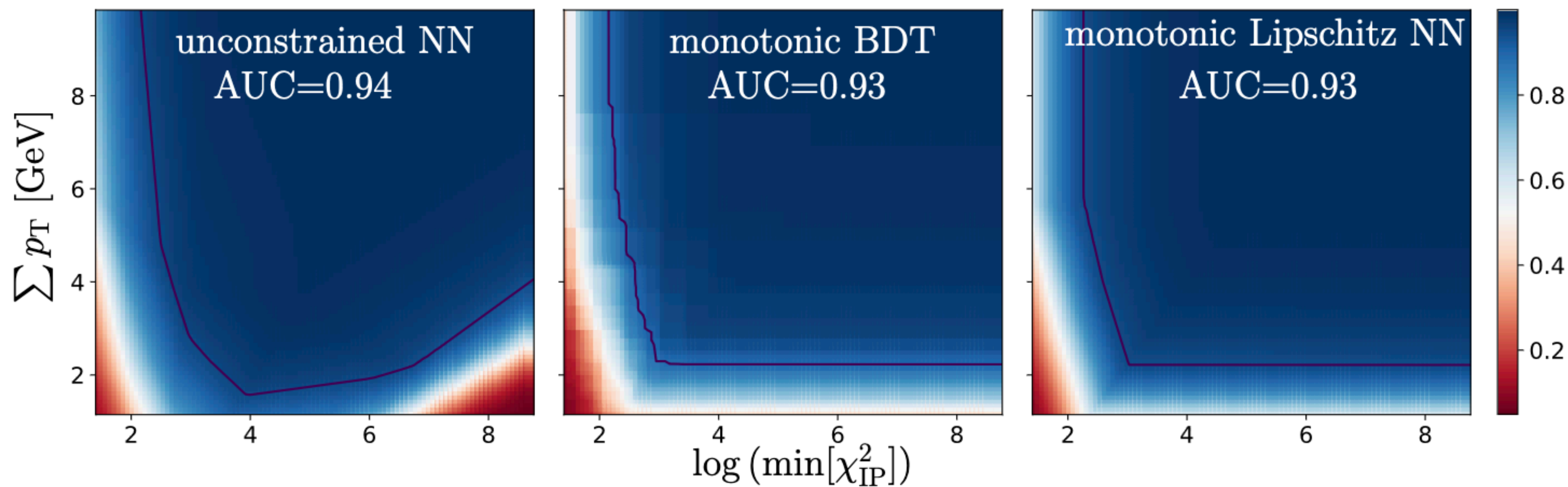
Monotonic Lipschitz Networks

This architecture is

1. provably robust
 2. provably monotonic
 3. universally approximating the target function class
 4. working well in practice
- Implemented in the LHCb trigger

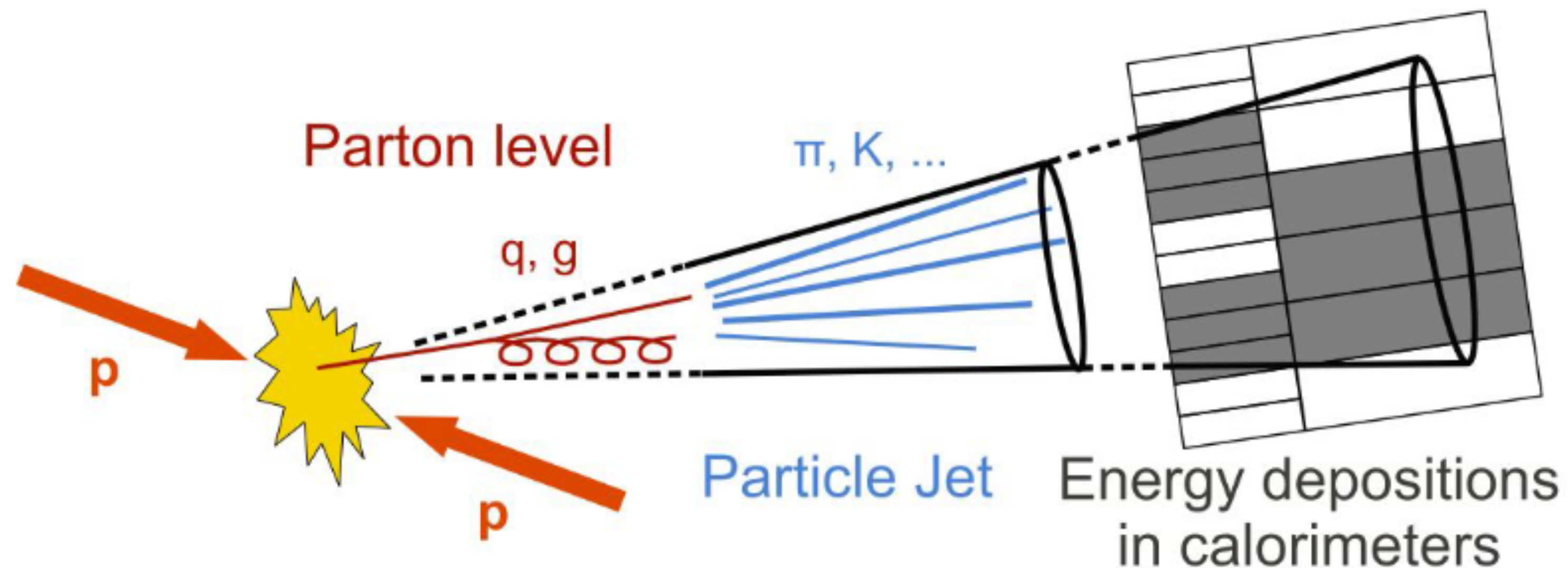


Example



Jet Tagging

Similar Task to Flavor Tagging, but for single jet



Input

16 Features of the Jet

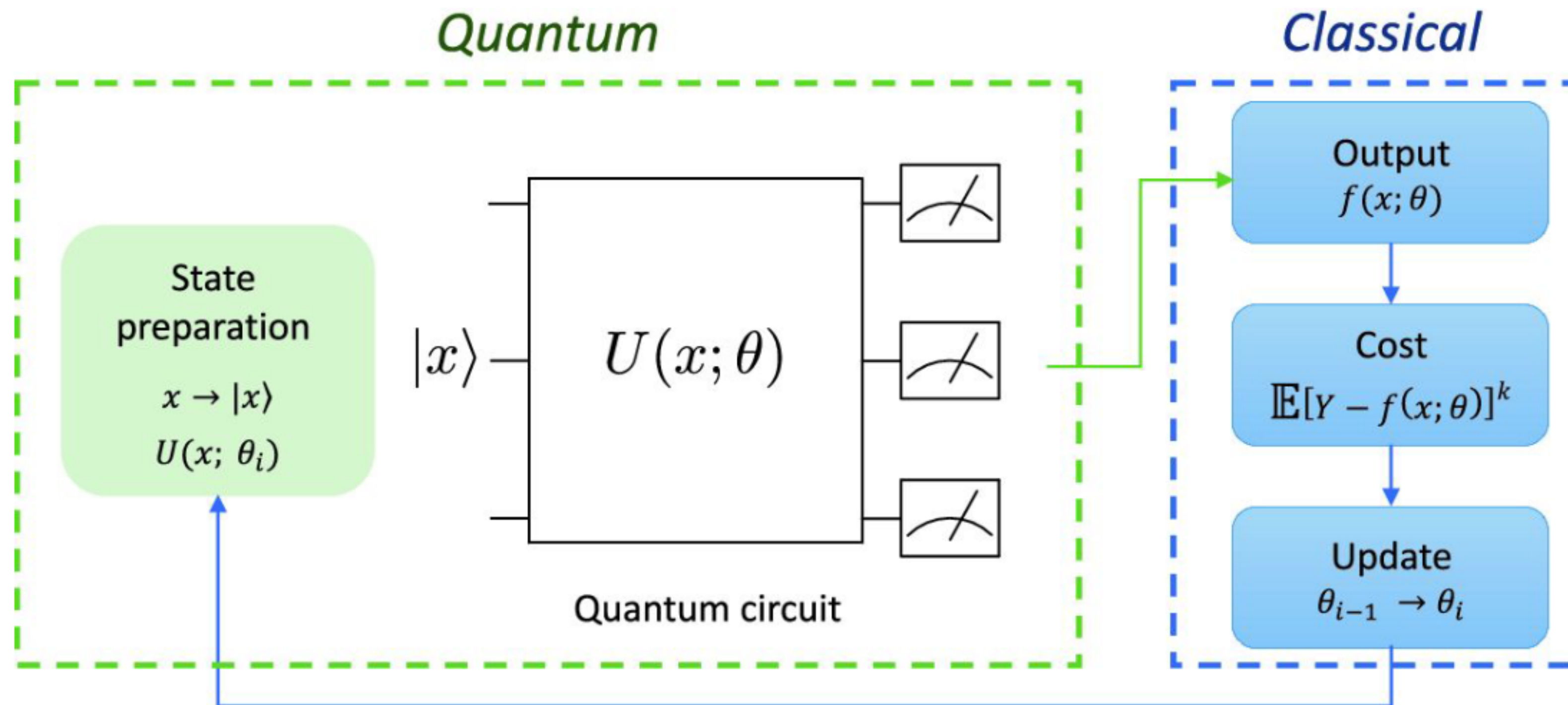
Output

Jet Tag (Binary)

Standard classification problem

Explorative study: QML for Physics

classification problem = **Variational Quantum Classifier**



Data are fed into variational quantum circuit.

Measurements of qubits are mapped to probabilities for labels.

Probabilities are used to estimate a cost function which is optimized through a classical optimizer

Competitive!

