Differentiable Programming for High Energy Physics

QCHS Stavanger

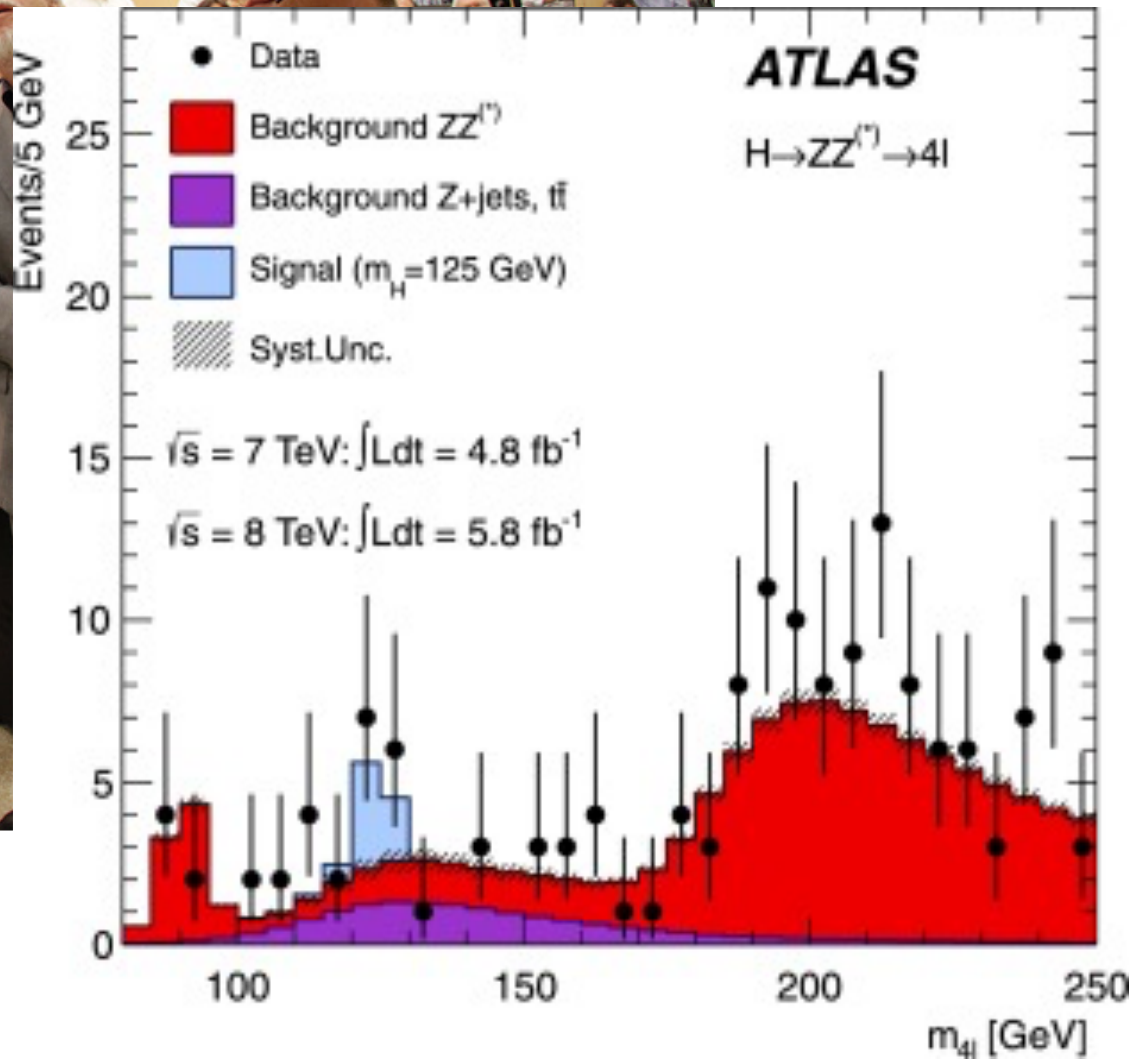
Lukas Heinrich

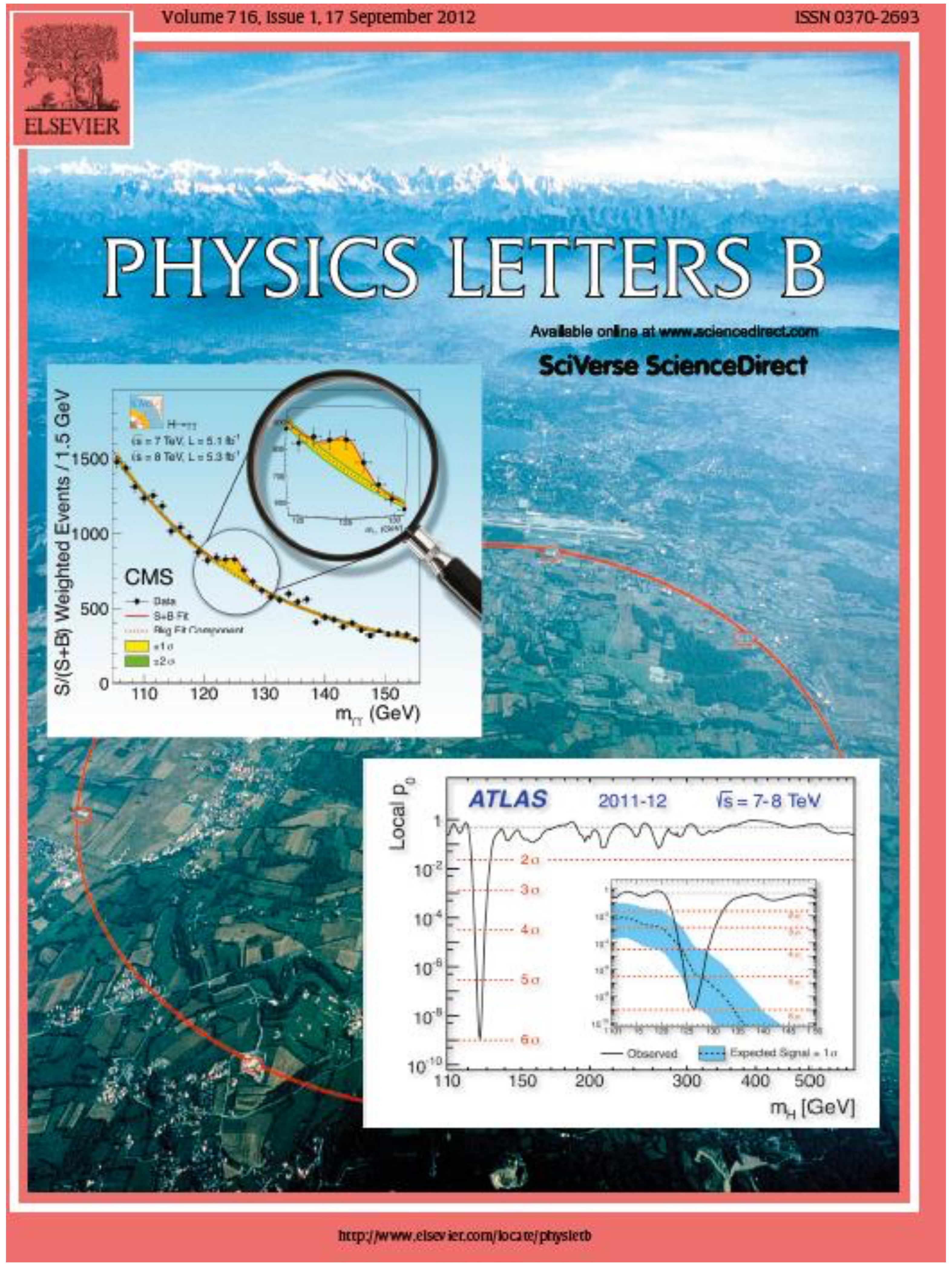
Technische Universität München

ODSL ORIGINS Data Science Lab

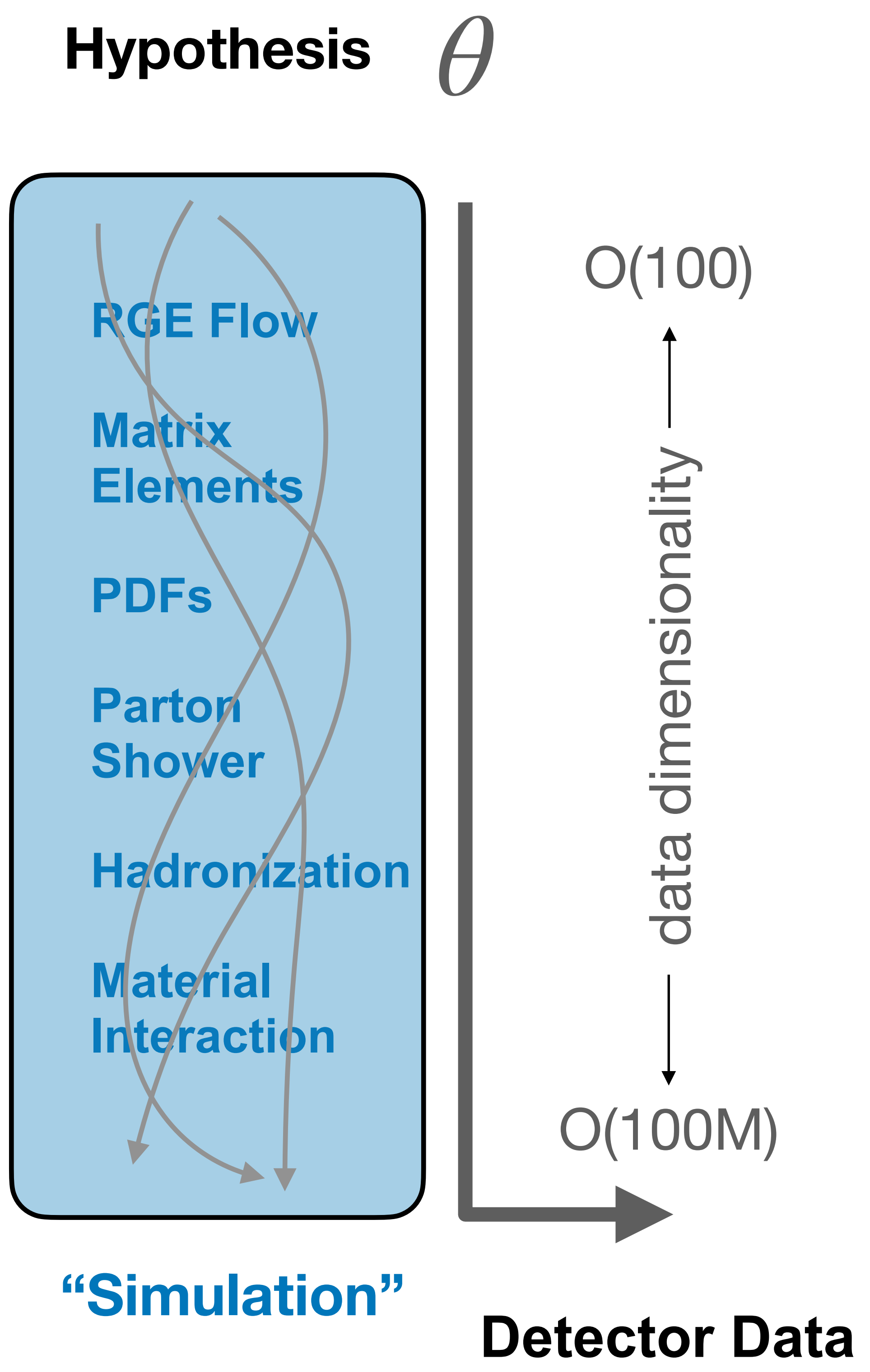
[img credit]

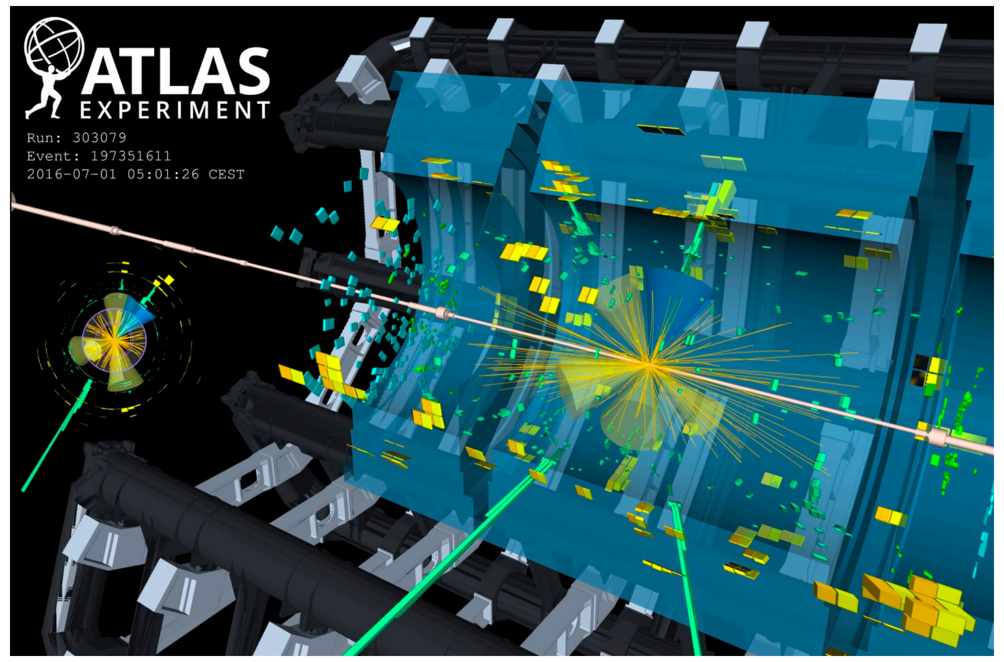
July 2012 in Physics





Inference in HNEP is fundamentally challenging





While we understand very well how our data is generated…

$$p(x, z|\theta) = p(x|z_\mathrm{d})p(z_\mathrm{d}|z_h)p(z_h|z_p)p(z_p|\theta)$$

…we can’t observe the intermediate states z:

$$p(x|\theta) = \boxed{\int \mathrm{d}z}\; p(x, z|\theta)$$

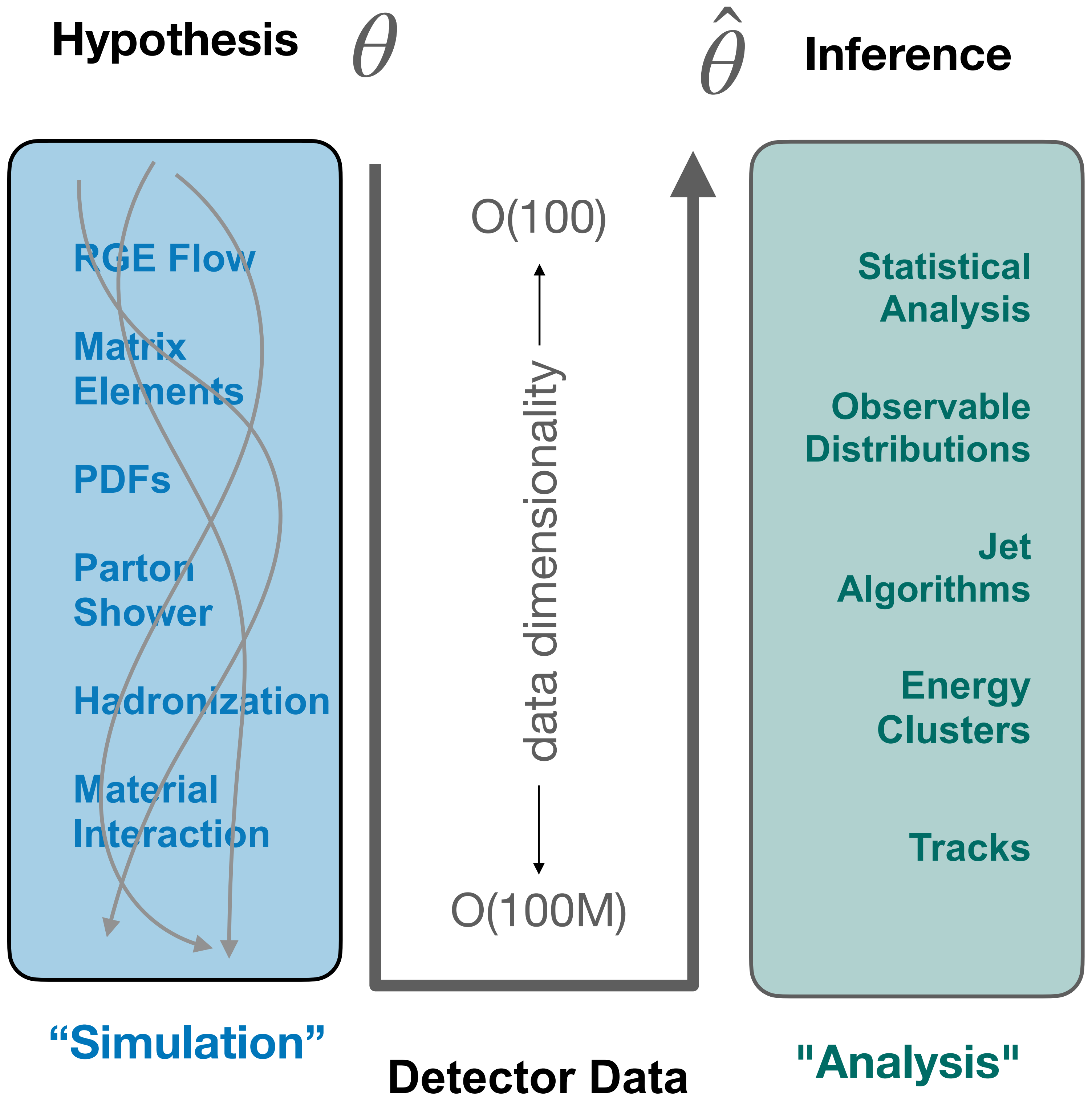
hopeless integral over millions of dim.

well-understood physics processes

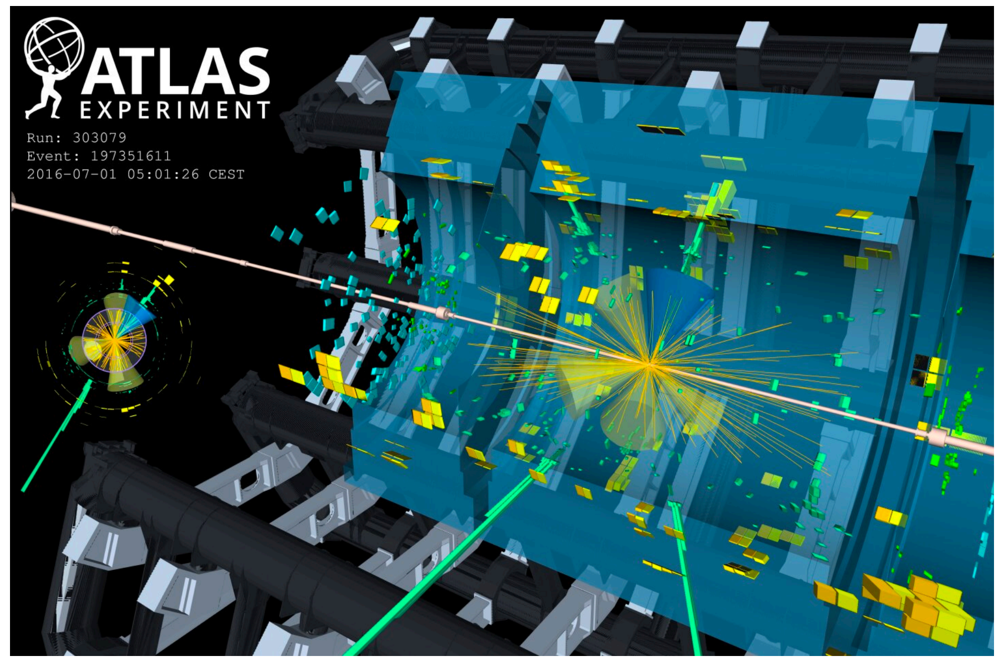
makes text-book data analysis impossible…

$$p(\theta|x) = \frac{p(x|\theta)}{p(x)}p(\theta)$$

Inference in HNEP is fundamentally challenging

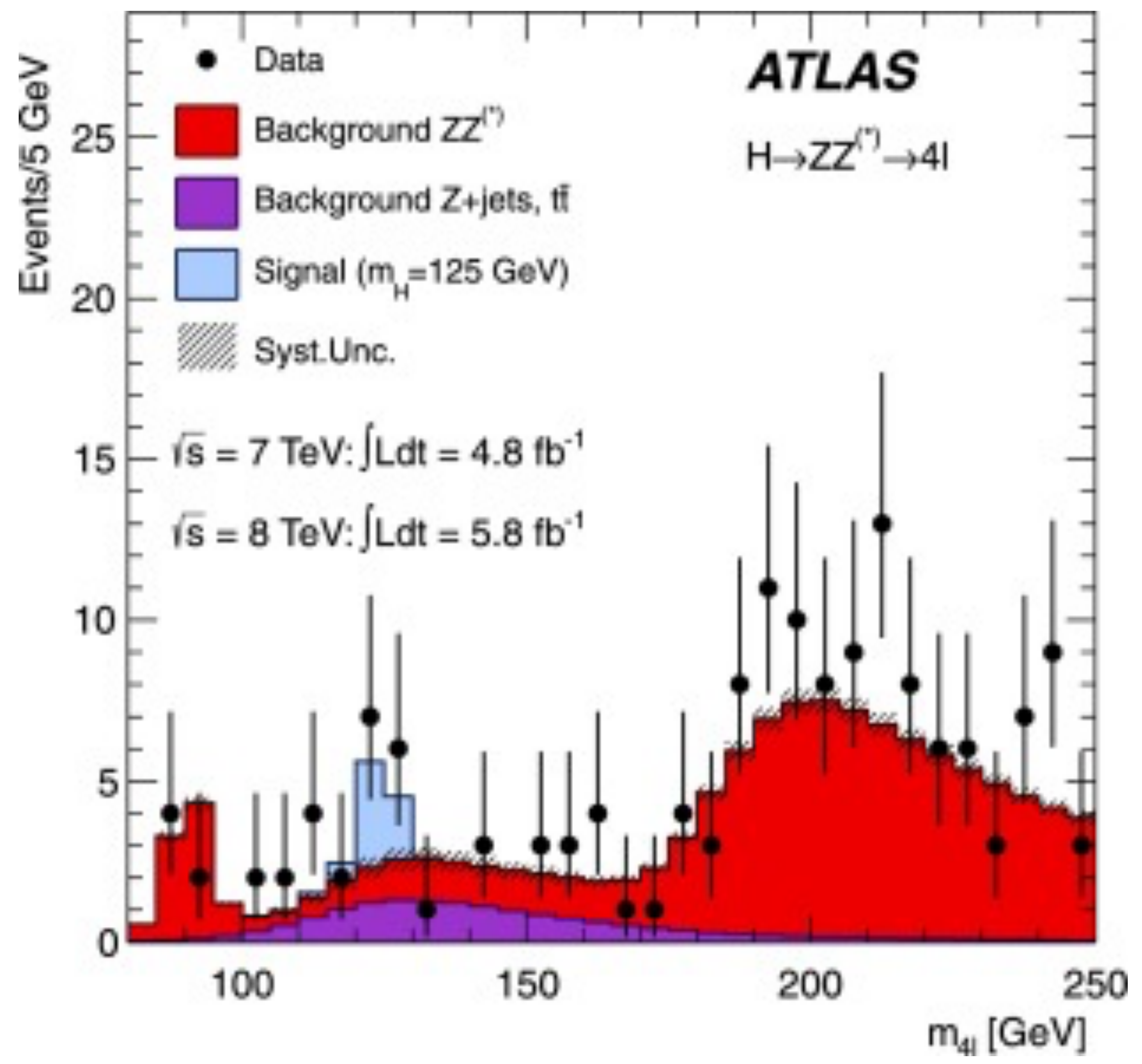


Detector Data



But we can generate sample data: $x \sim p(x \,|\, \theta)$ by encoding our physics into **(very costly) simulators**

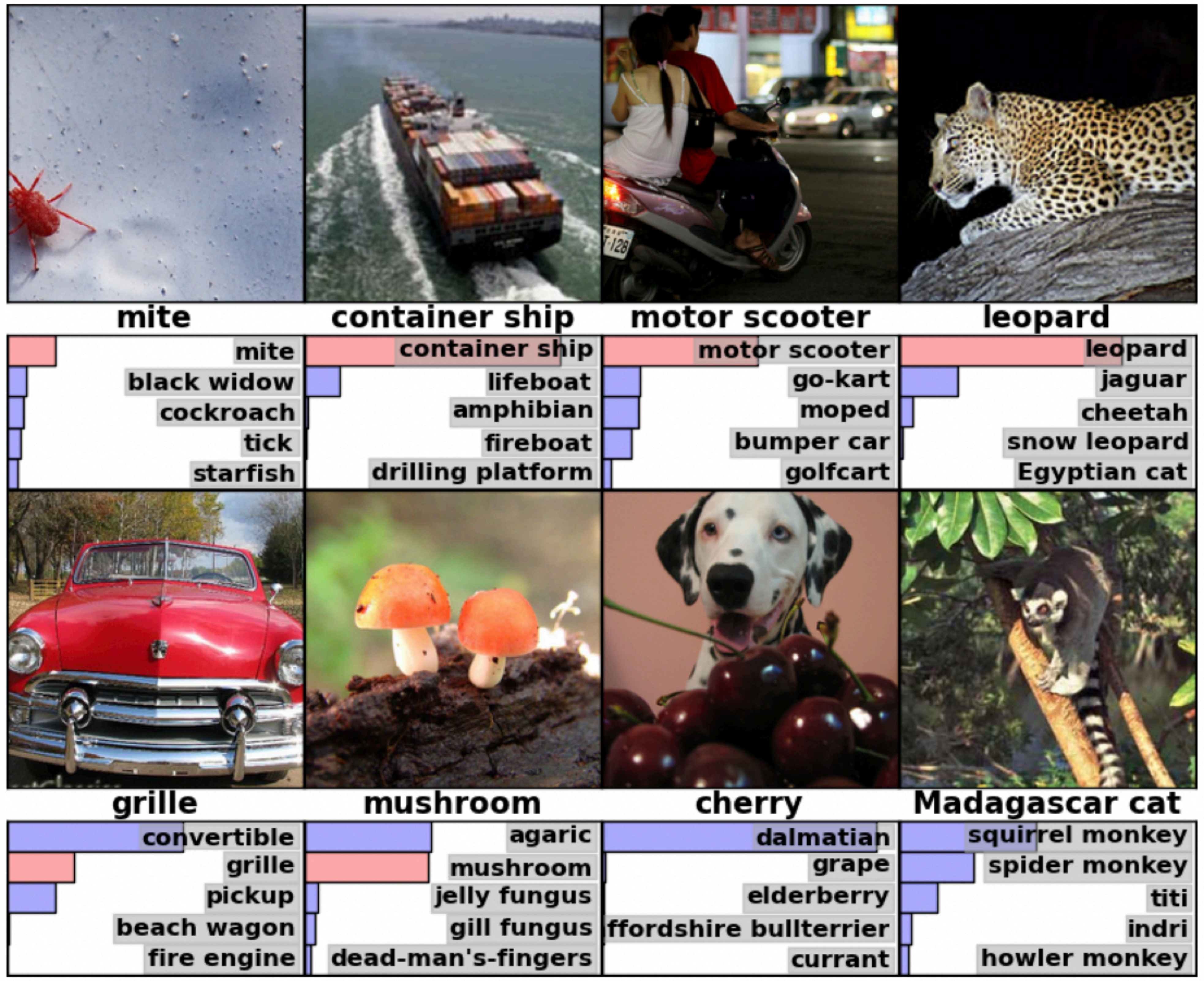
Common Strategy: try to find a good low-dimensional observables: $x \to y_x = f(x)$



estimate $p(y_x \,|\, \theta)$ from samples for inference i.e. $p(\theta \,|\, y_x)$

This is the core of **“reconstruction” and “analysis”**

July 2012 in Computer Science



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Breakthrough of neural-network based "Deep Learning"

Impressive Progress in the last Decade

"A painting by Grant Wood of an astronaut couple, american gothic style"

Language

"This is a picture of Barack Obama"

"His foot is positioned on the right side of the scale"

"The scale shows a higher weight"

generate low-level data from high-level concepts

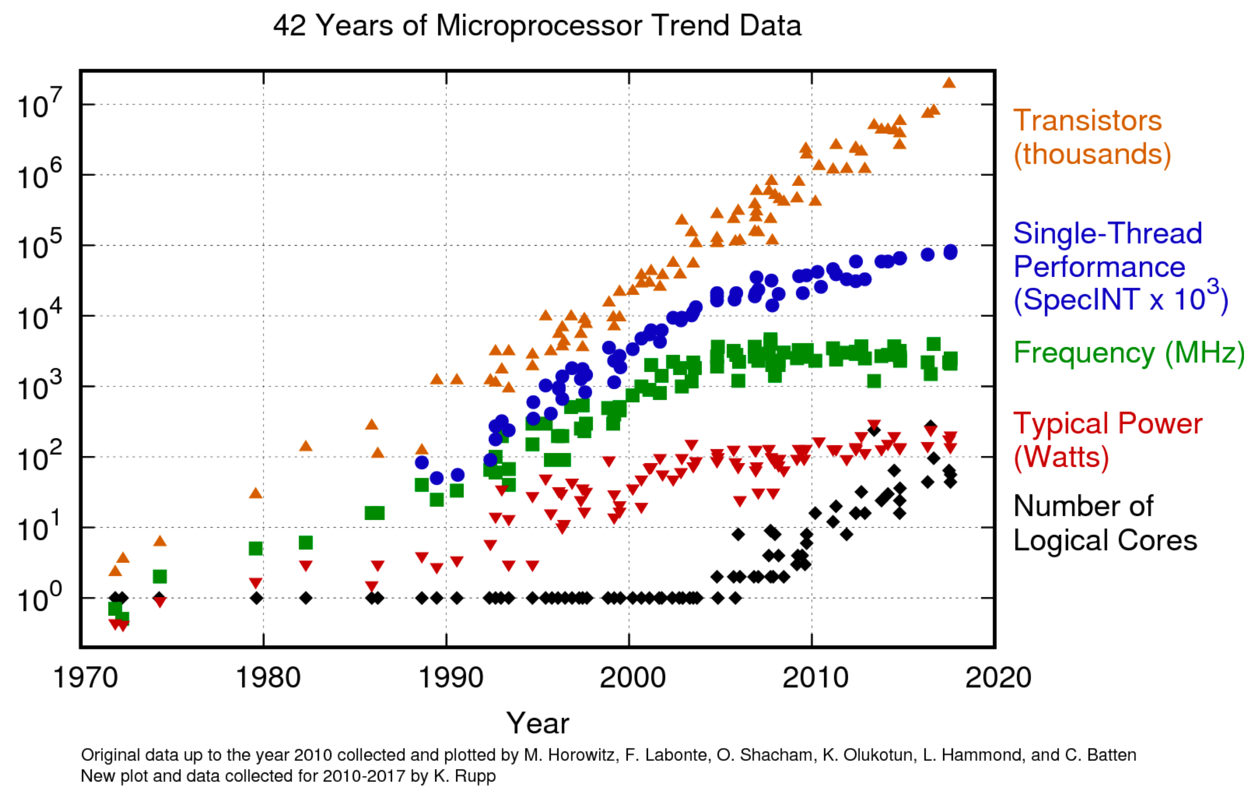
reconstruct high-level concepts from raw data

Pixels

ML Opportunities in Fundamental Physics

Acceleration of Computation

(e.g. sometimes by searching for a good approximation)



turing lecture

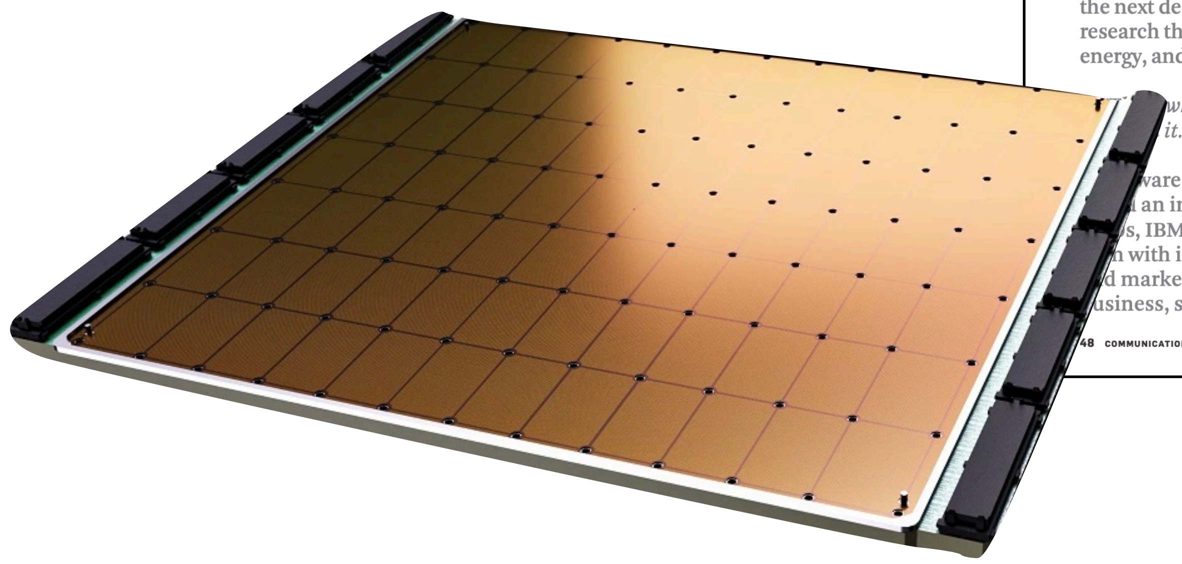
DOI:10.1145/3282307

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

BY JOHN L. HENNESSY AND DAVID A. PATTERSON

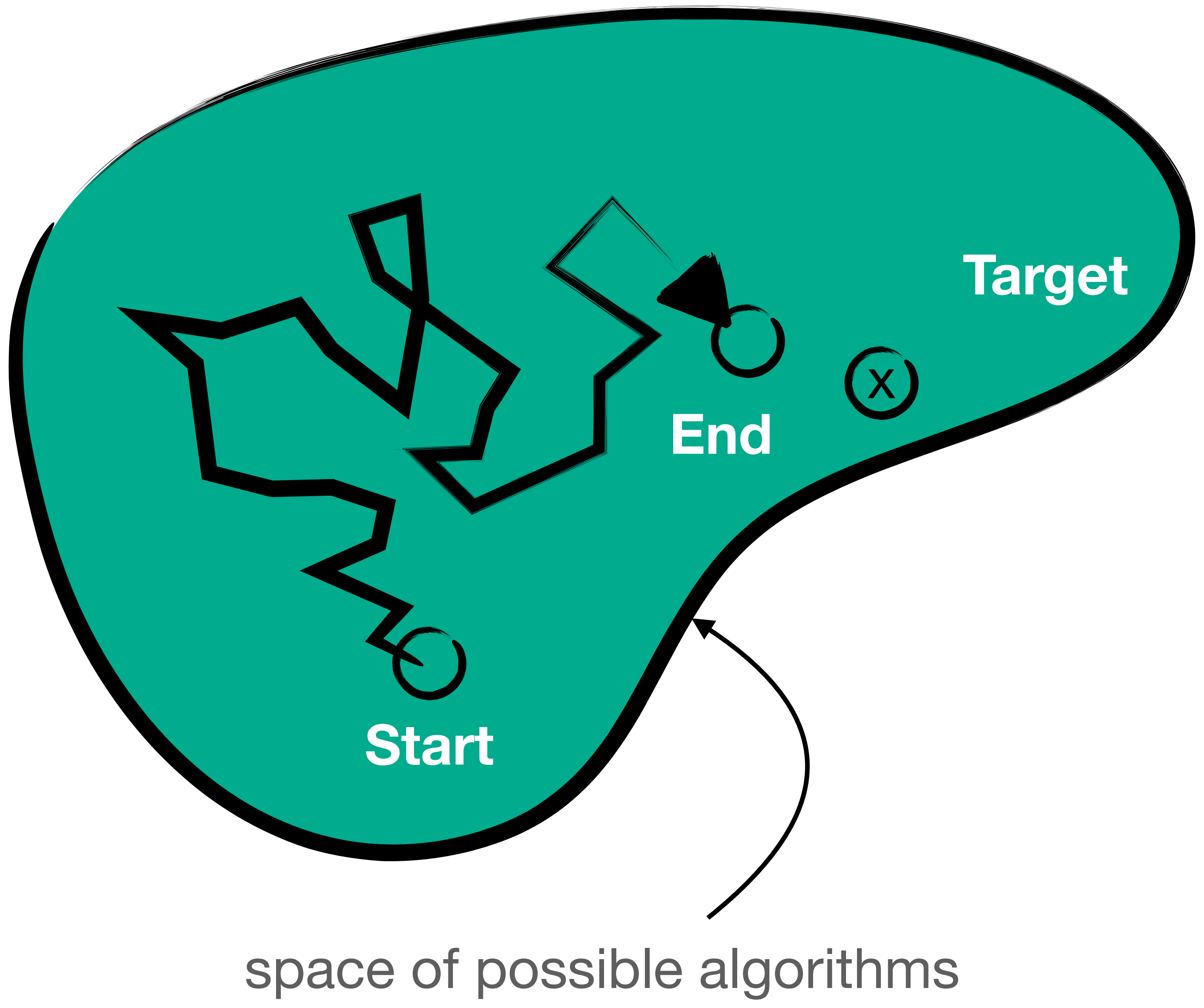
A New Golden Age for Computer Architecture

WE BEGAN OUR Turing Lecture June 4, 2018[11] with a review of computer architecture since the 1960s. In addition to that review, here, we highlight current challenges and identify future opportunities, projecting another golden age for the field of computer architecture in the next decade, much like the 1980s when we did the research that led to our award, delivering gains in cost, energy, and security, as well as performance.



Search for new (better) Algorithms

(e.g. targeted search based on samples)



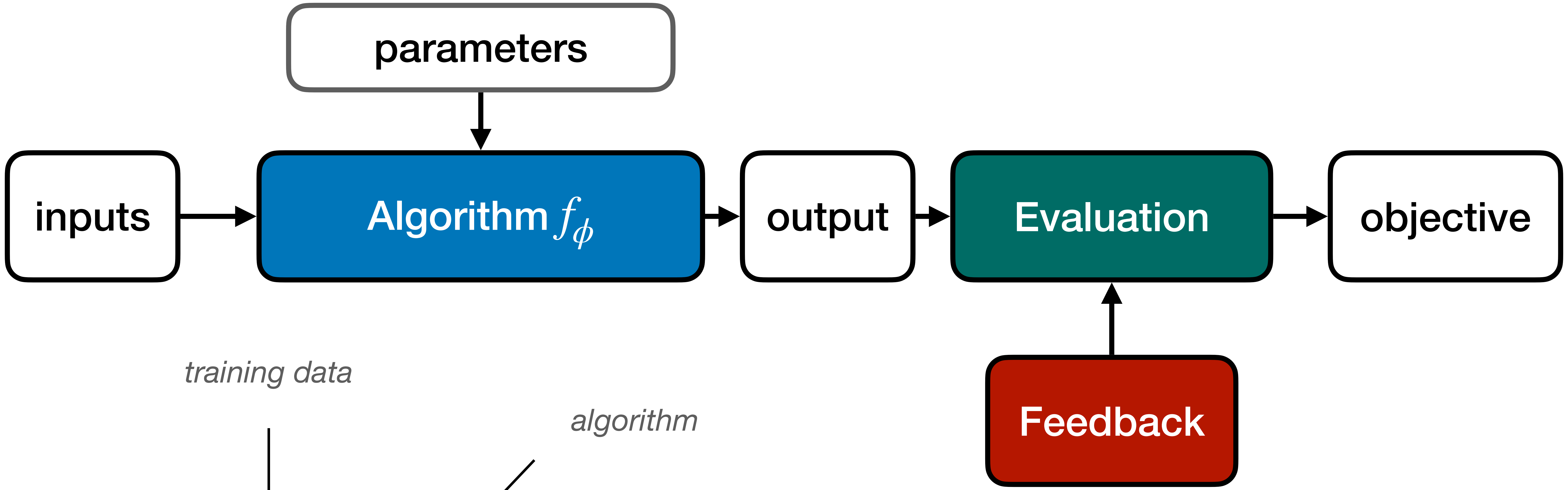
simulation side: the physics is fixed: nothing to search for →speed up simulation

up to us to find best observables →search for best reconstruction

Lightning Summary of ML

Learning: data-driven search for a function with optimal performance in a huge

Space of Algorithms



training data

algorithm

Performance: $$\mathbb{E}_{(x,y)}\mathcal{L}(f_\phi(x), y)$$

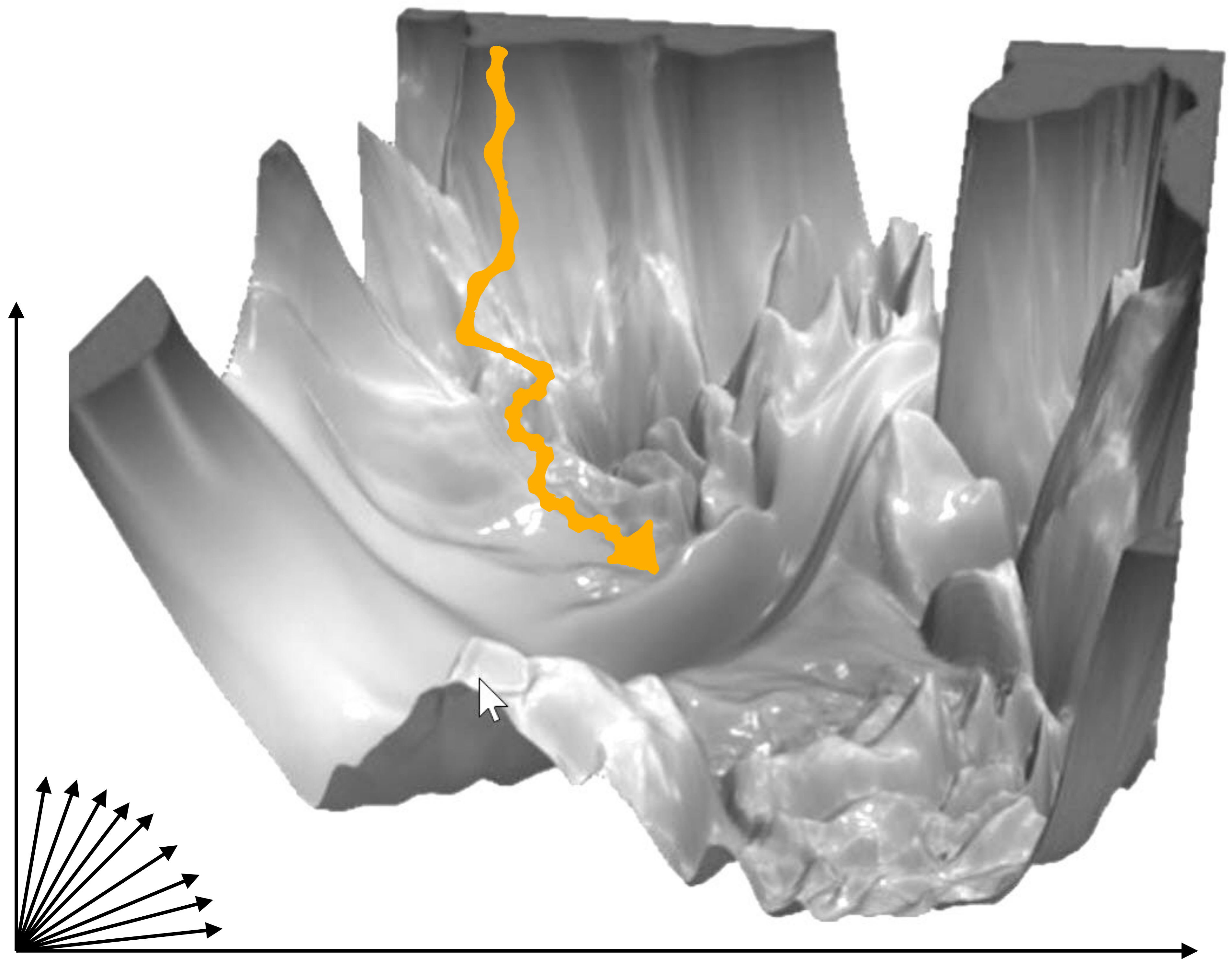
performance evaluation

How do we learn practically?

Lightning Summary of ML

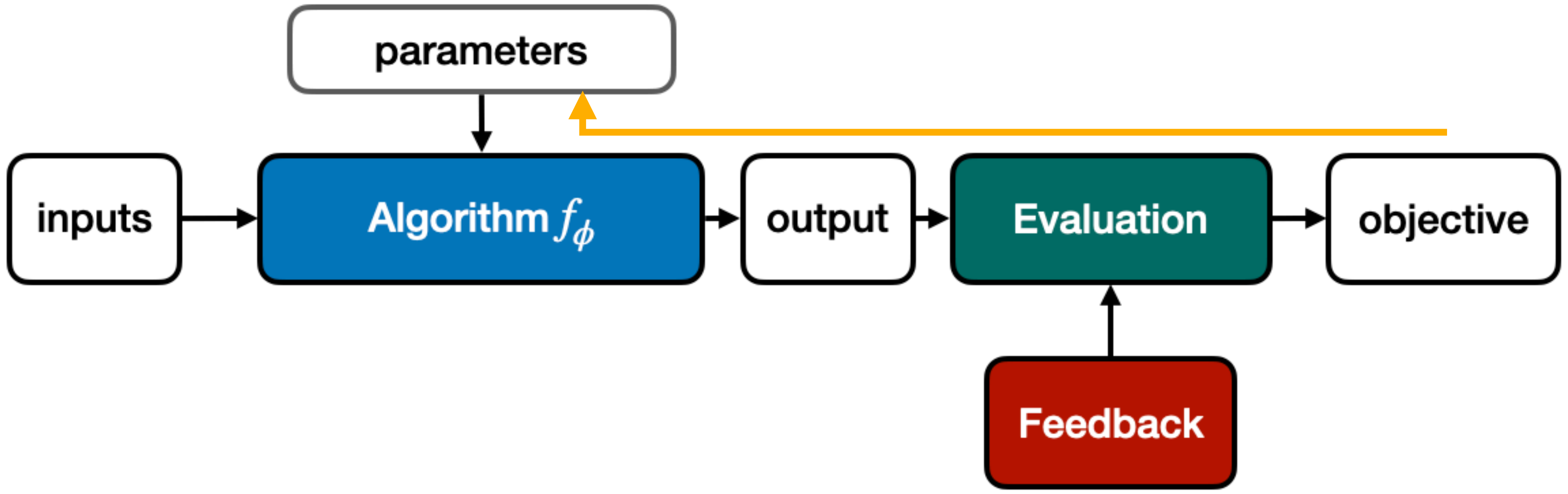
search space should be large enough → trillions of parameters! How could this work?

→ **gradient-based optimization ("good sense of direction")**



To deal with hyper-planes in a 14-dimensional space, visualize a 3D space and say 'fourteen' to yourself very loudly. -Hinton (DL pioneer)

$$\frac{\partial L}{\partial \phi} = \frac{\partial L}{\partial f}\frac{\partial f}{\partial \phi}$$

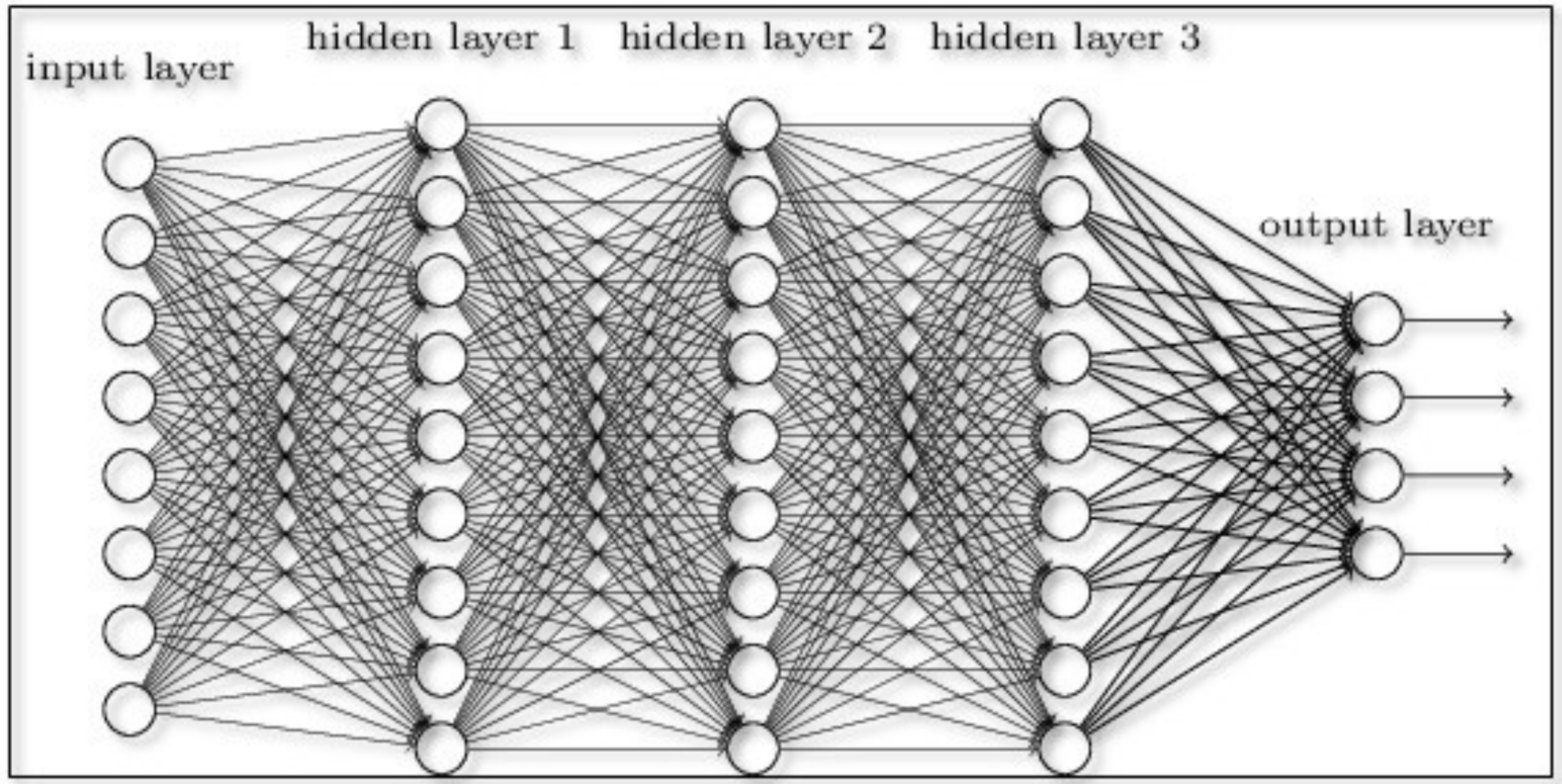


→ requires **algorithms** and **evaluation** to **be differentiable**

Finding the right Search Space

At first

fixed but generic, large and **easily differentiable** function class:



manual derivation of efficient gradient computation

Increasingly

domain-specific, arbitrary computation encoding e.g. symmetries, dynamics, ...

$$R_g y = f(R_g x) \qquad \dot{x} = f(x)$$

[M. Bronstein]

?

Differentiable Programming

The key: **programming languages** whose programs are **inherently differentiable**

- ***avoid overhead*** of computer algebra (symbolic differentiation)
- ***exact gradients*** instead of numerical approx. (unstable in high dimensions)

```python
import jax
import jax.numpy as jnp

def func(x):
  y = x
  for i in range(4):
    y += x[0]**2 + jnp.sin(x[1]) + jnp.exp(-x[2])
  y = y.sum()
  return y
```

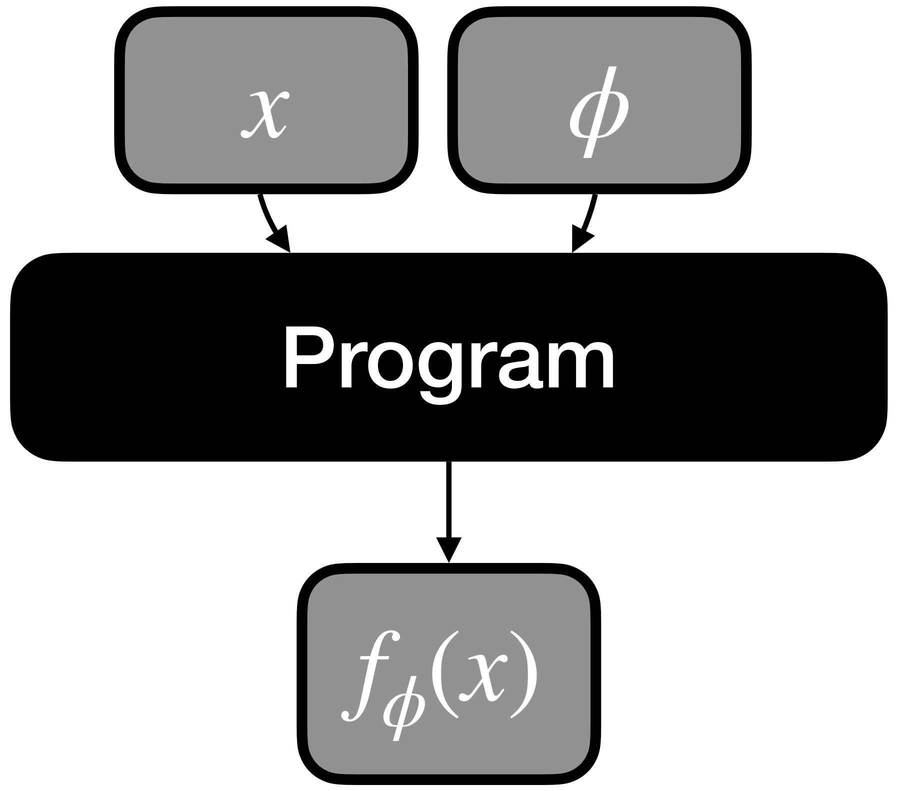
exact gradients!

```python
gfunc = jax.value_and_grad(func)
gfunc(jnp.array([2.,3.,-2]))
```

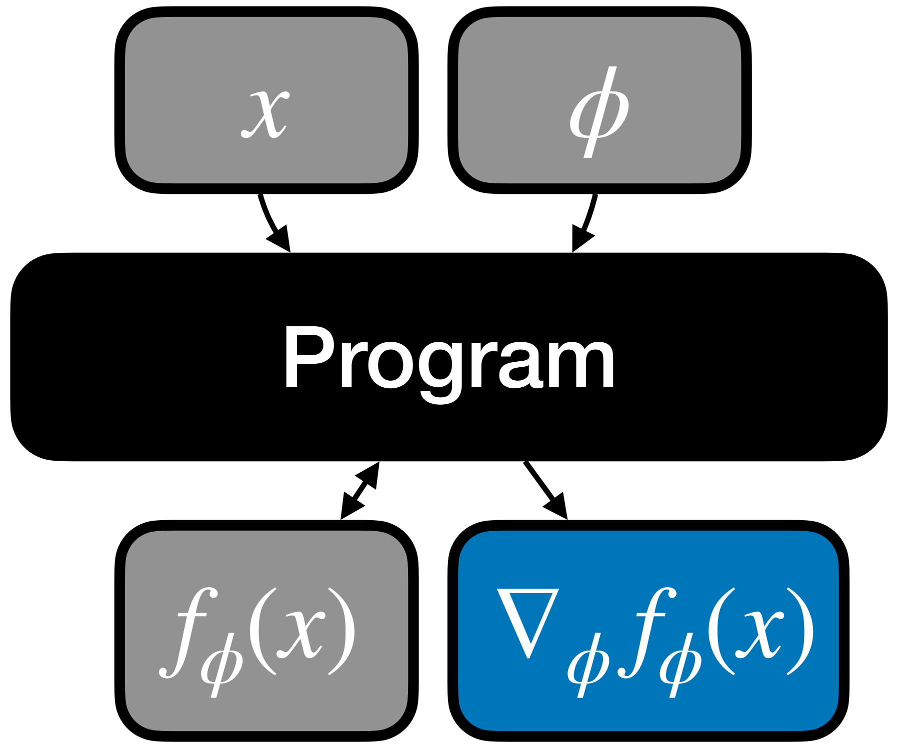
```
(DeviceArray(141.36212, dtype=float32),
 DeviceArray([ 49.        , -10.8799095, -87.66867   ]
```

TensorFlow JAX PYTORCH

… but also C++, Fortran, … (see backup)



standard programming

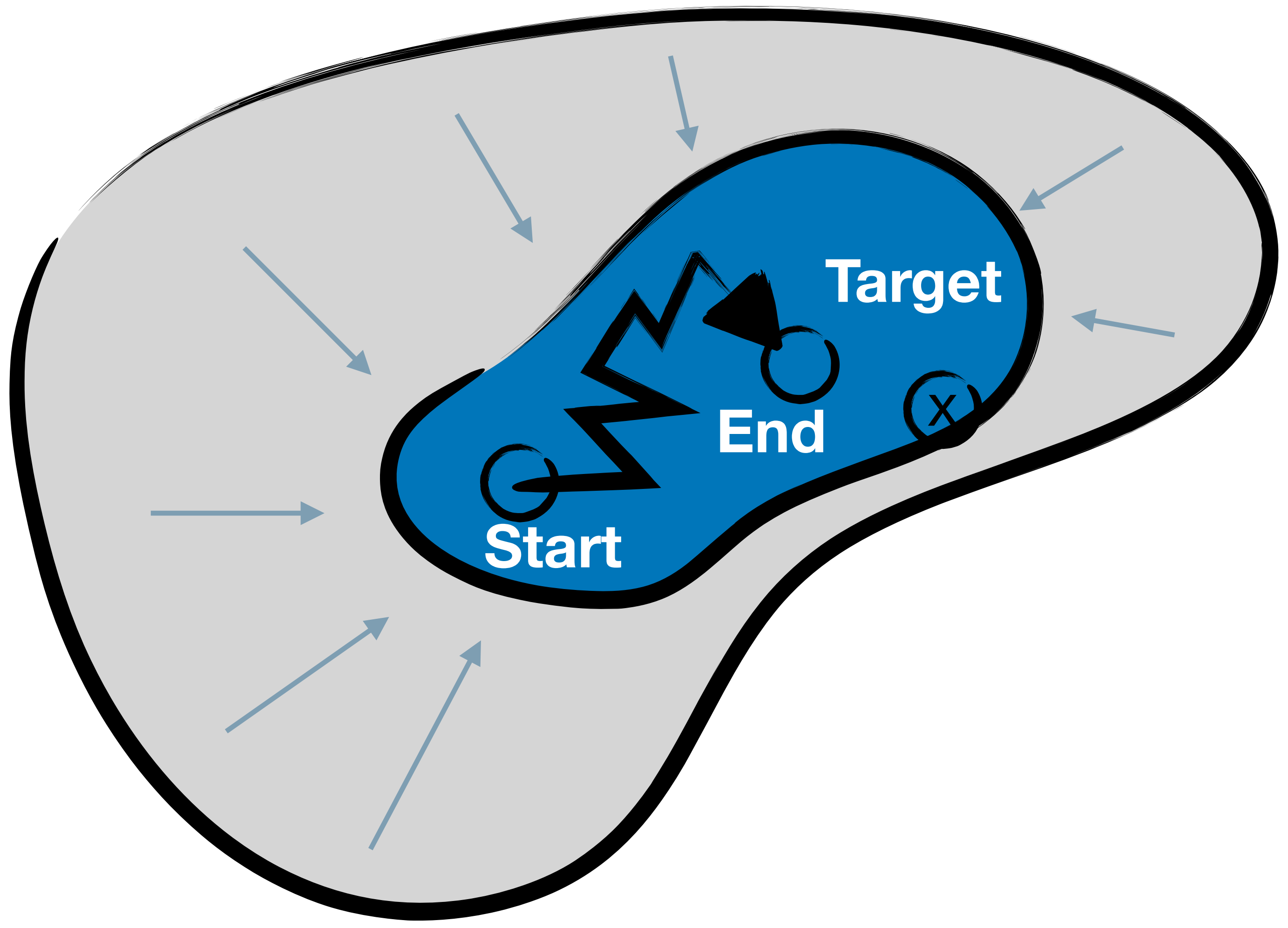


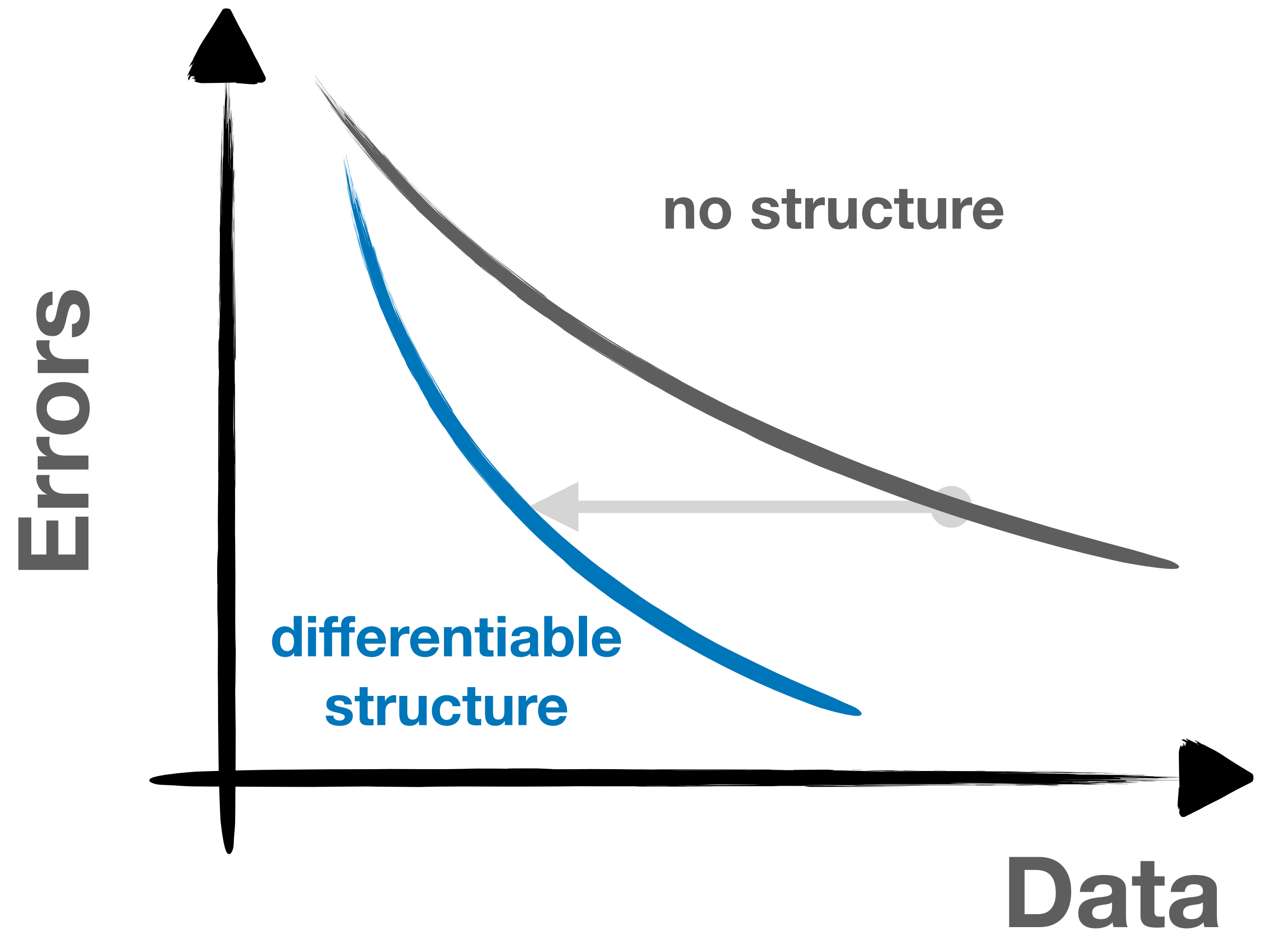
differentiable programming

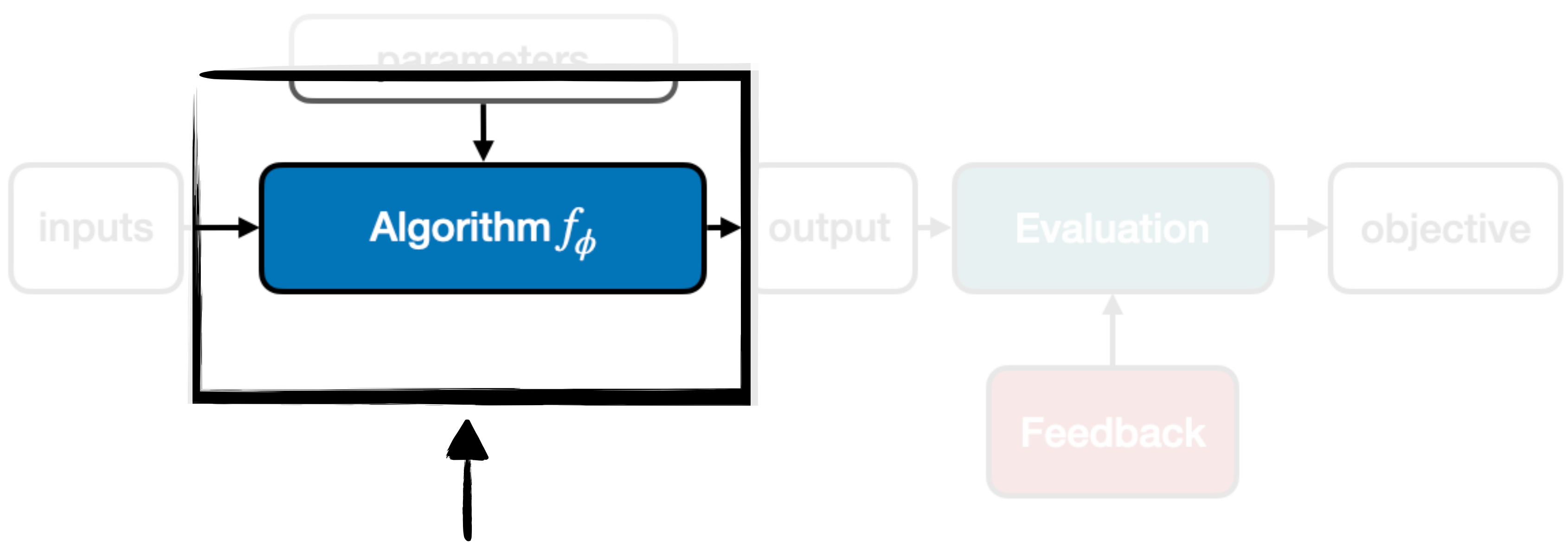
Differentiable Programming in ML

Immediate Gains from DiffProg: allows us to add physics into ML models

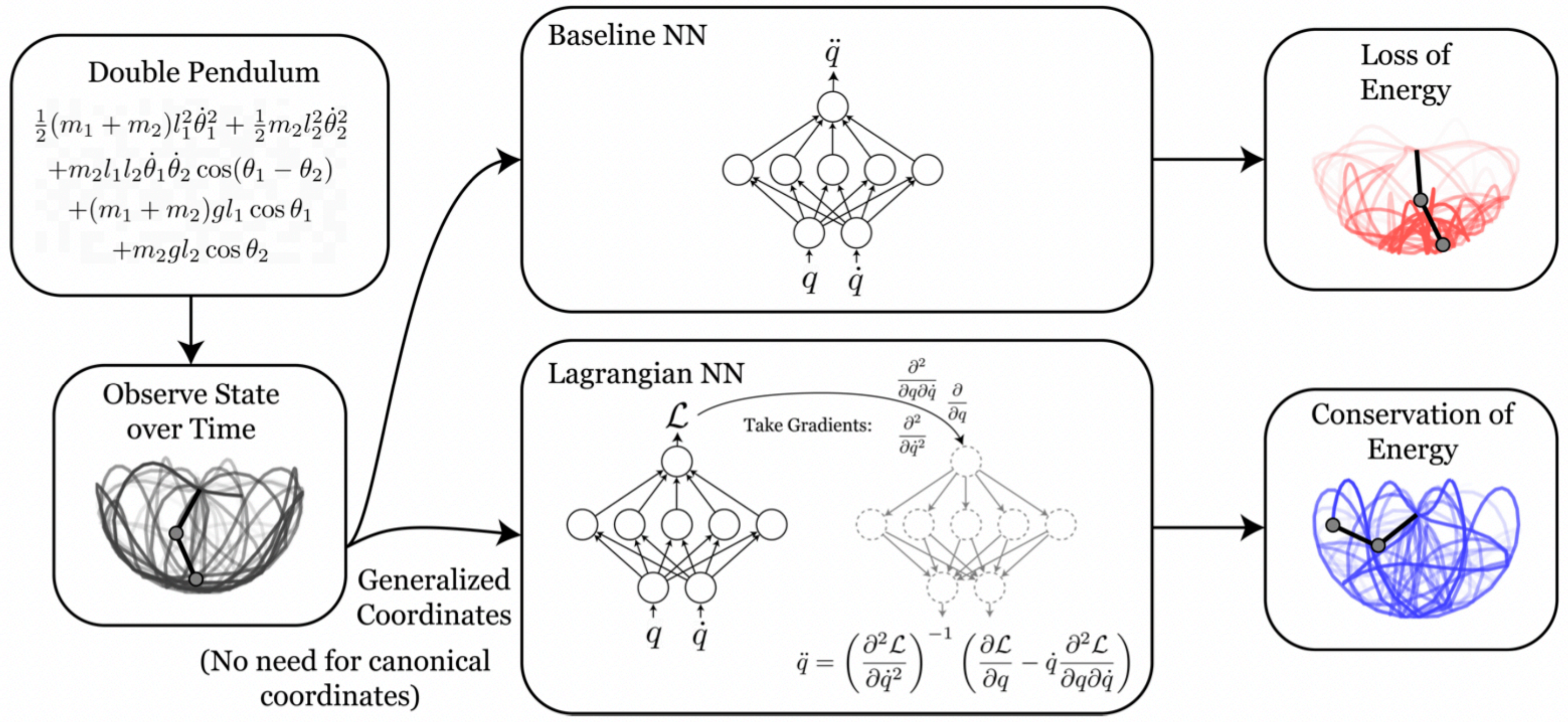
- bias towards good solutions by constraining solution space
- hard-coded knowledge does not need to be learned from data (efficiency)



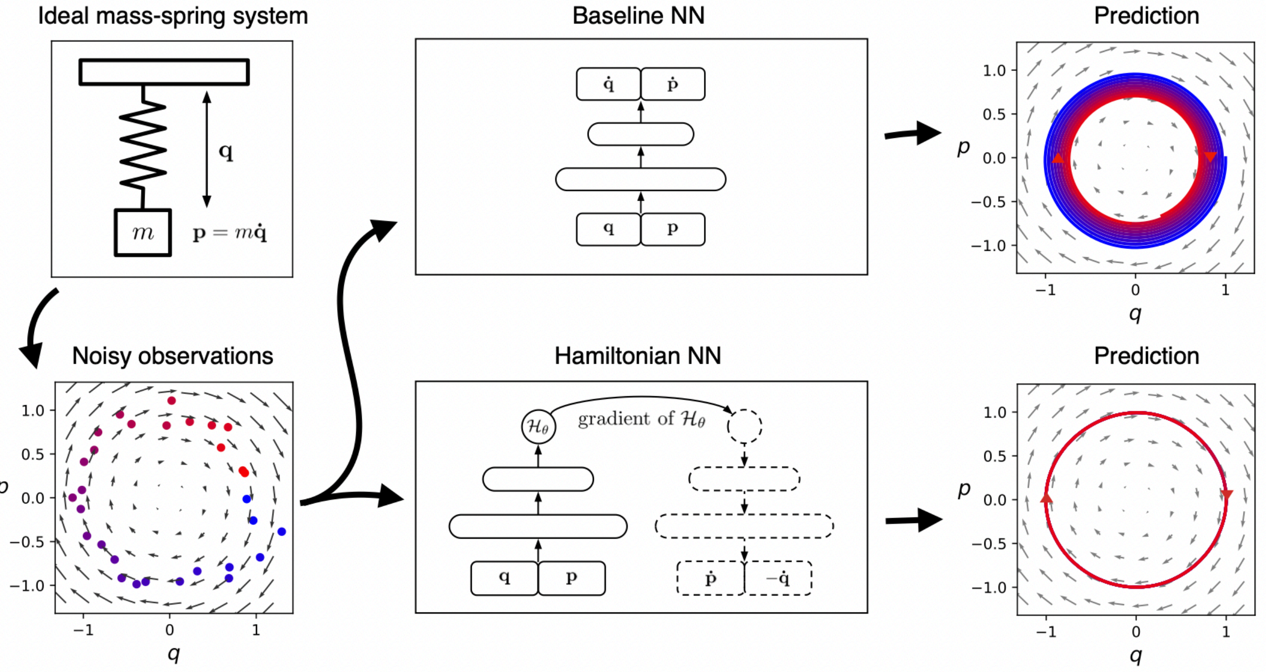




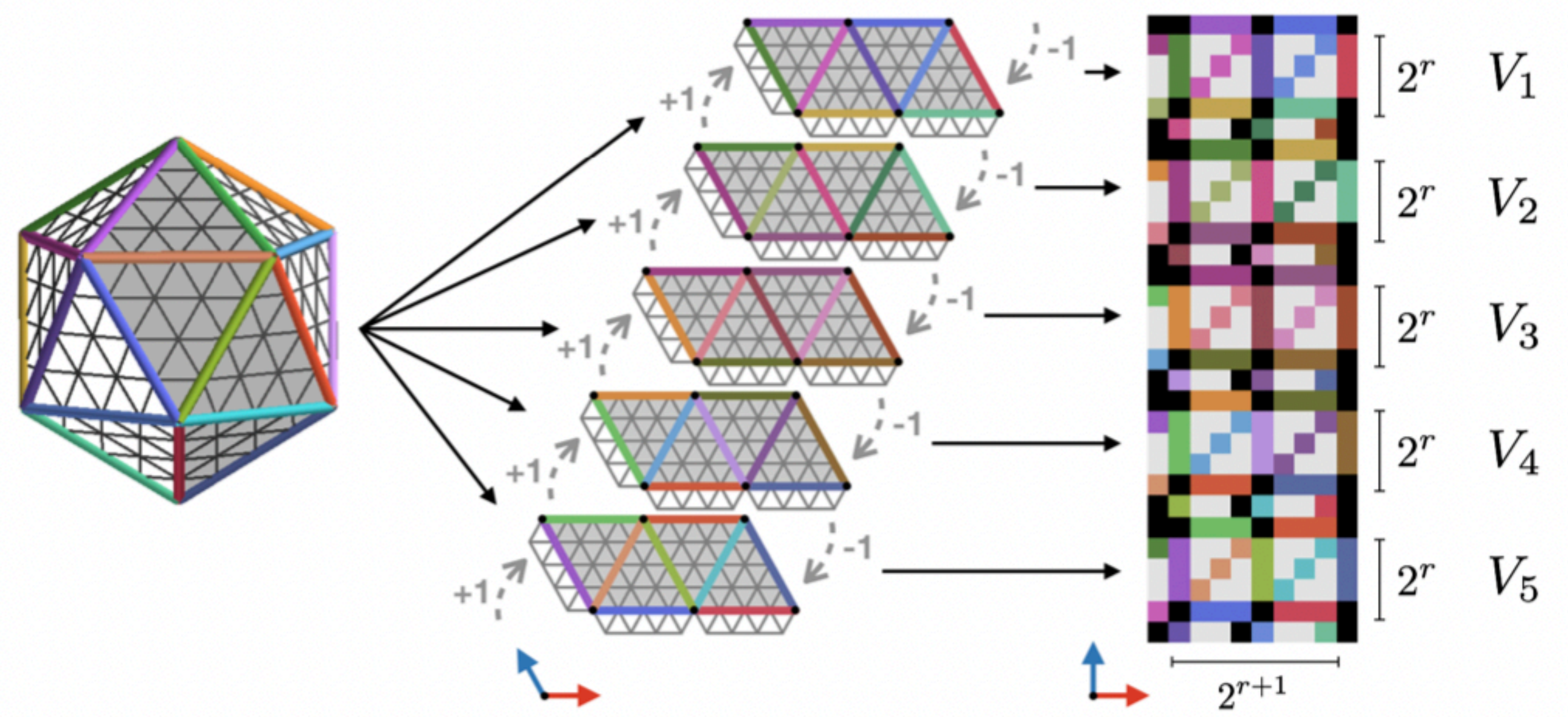
Differentiable Programming in ML



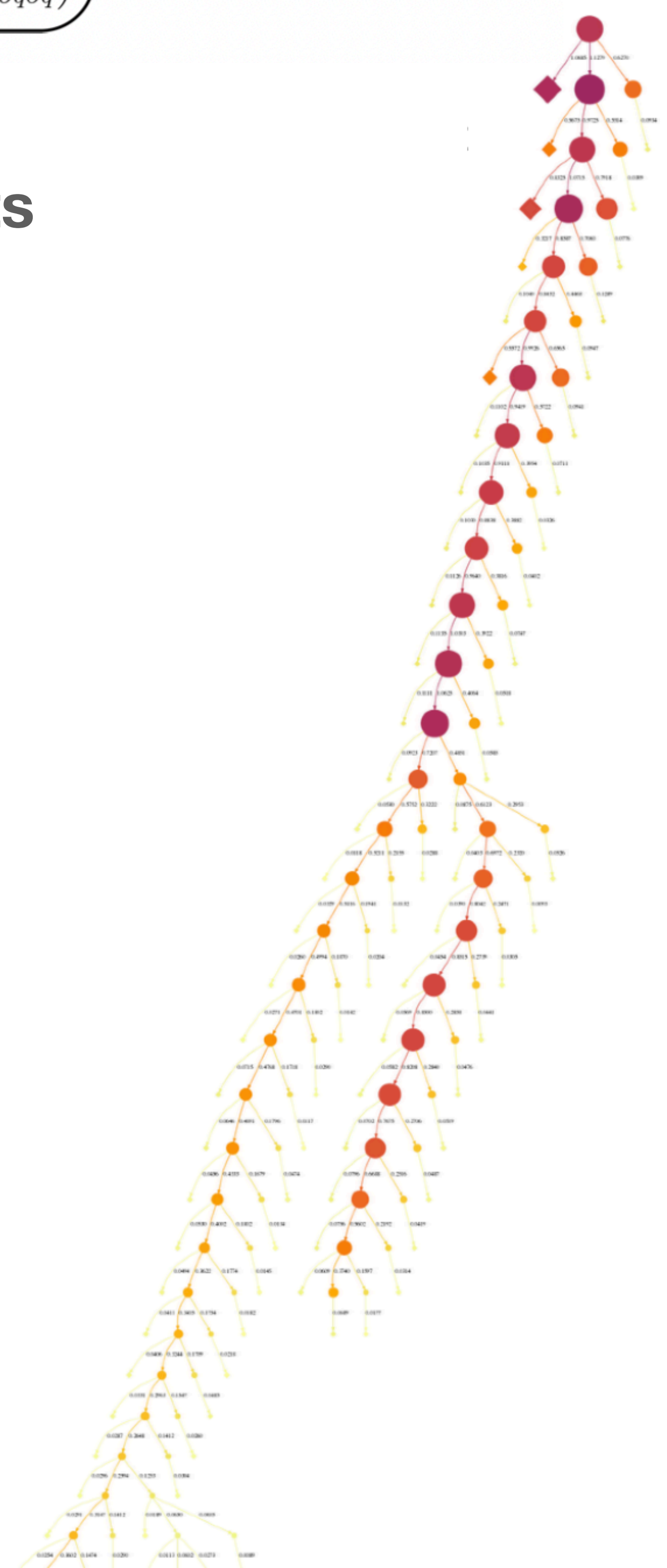
Lagrangian Neural Nets
arXiv: 2003.04630



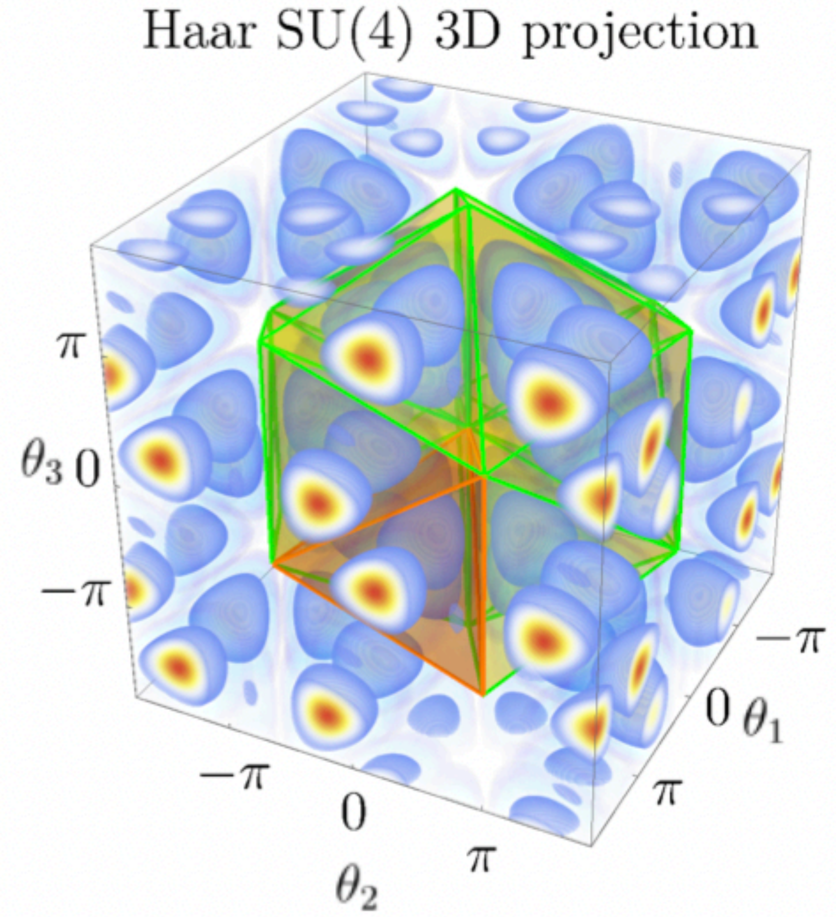
Hamiltonian Neural Nets
arXiv:1906.01563



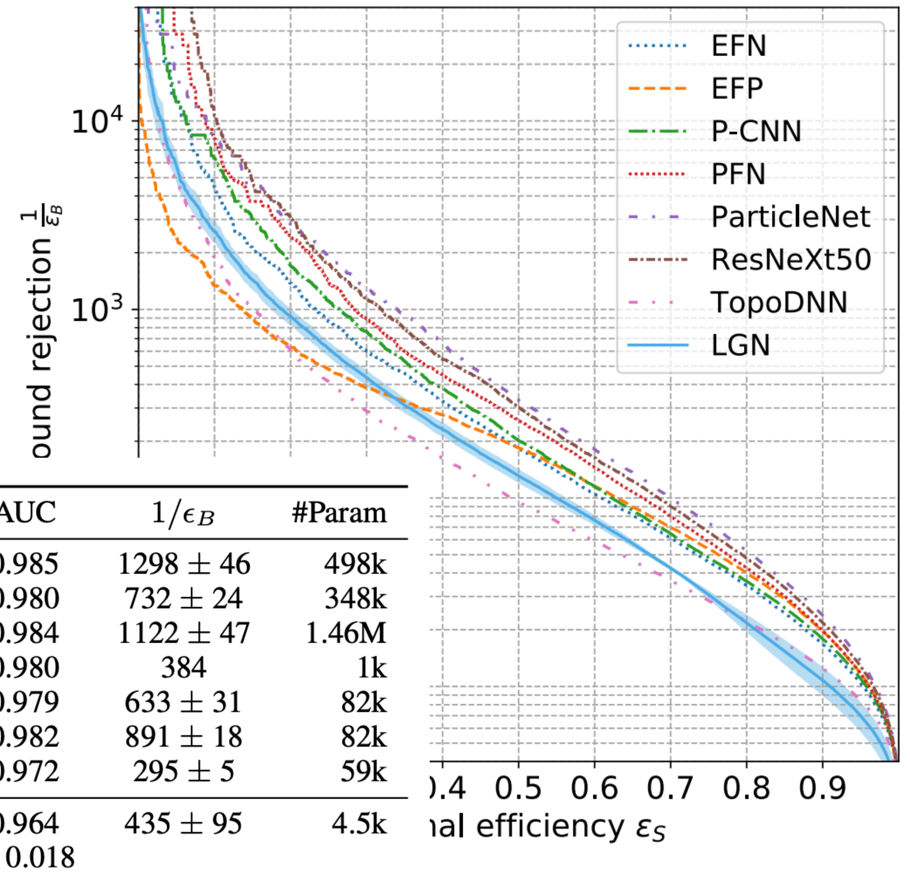
Gauge-Equivariant Convolutional Neural Networks



Neural Nets with QCD-like Structure
arXiv:1702.00748



SU(N)-Equivariant Normalizing Flows

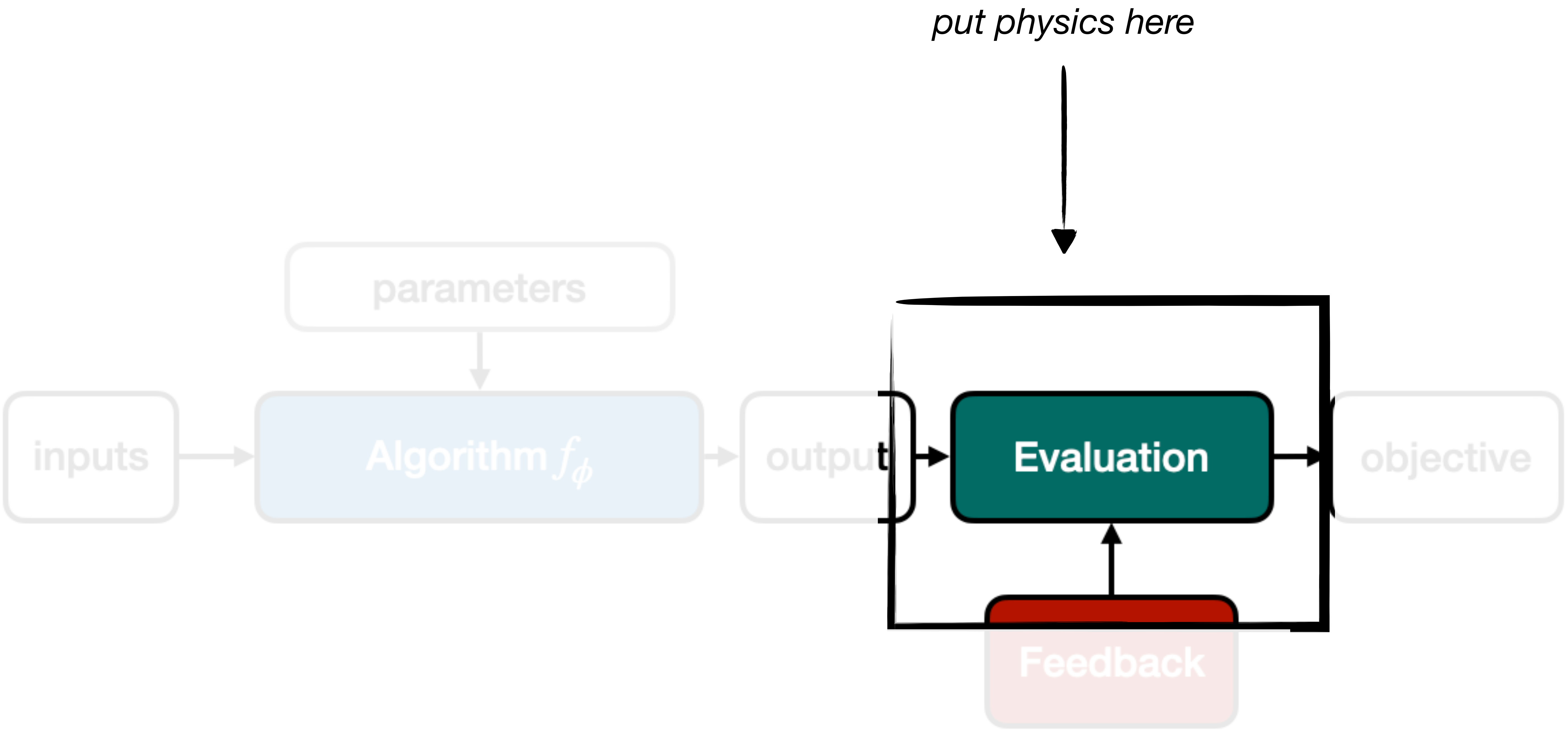


Architecture	Accuracy	AUC	$1/\epsilon_B$	#Param
ParticleNet	0.938	0.985	1298 ± 46	498k
P-CNN	0.930	0.980	732 ± 24	348k
ResNeXt	0.936	0.984	1122 ± 47	1.46M
EFP	0.932	0.980	384	1k
EFN	0.927	0.979	633 ± 31	82k
PFN	0.932	0.982	891 ± 18	82k
TopoDNN	0.916	0.972	295 ± 5	59k
LGN	0.929 $\pm$ .001	0.964 $\pm$ 0.018	435 ± 95	4.5k

Lorentz-Invariance
arXiv:2006.04780

Differentiable Programming in ML

Complementary Approach: add physics-driven *evaluation*

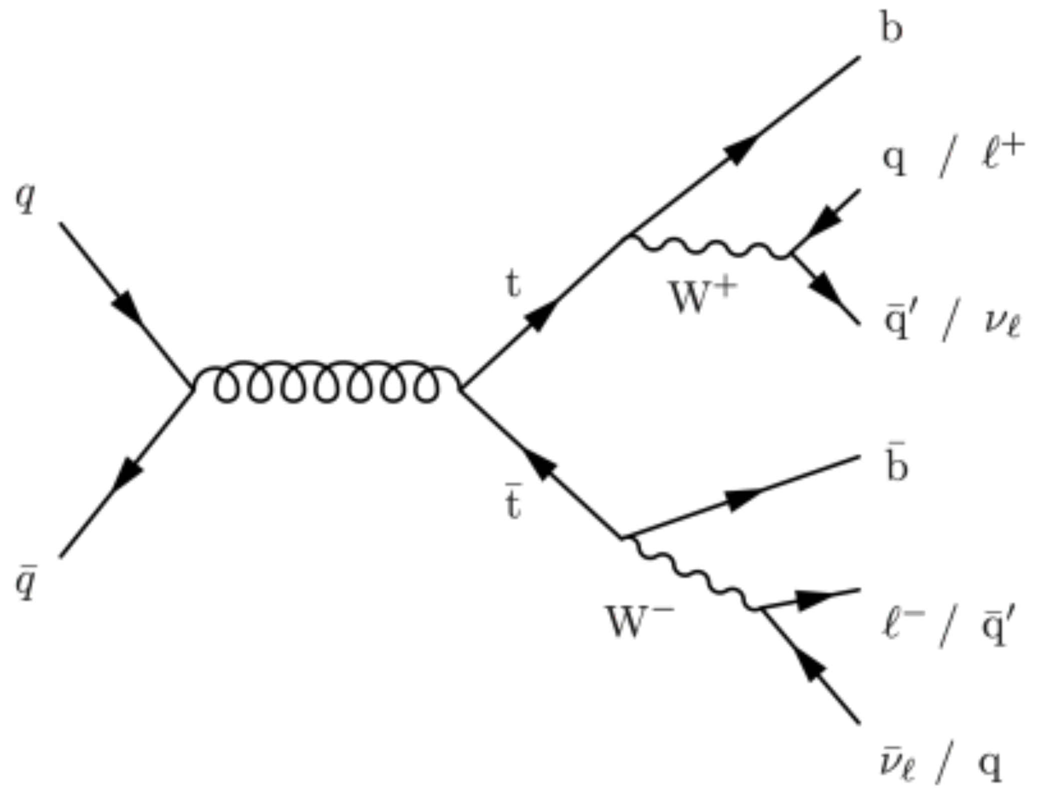


Differentiable Programming in ML

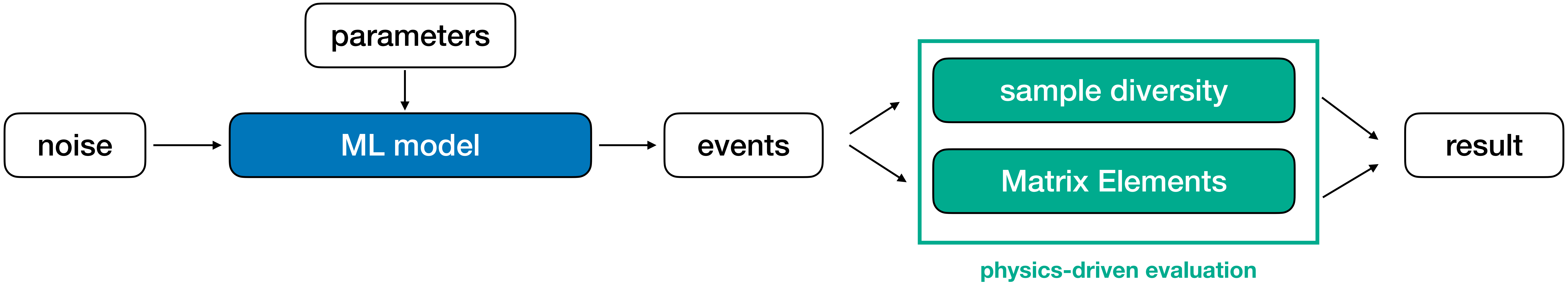
Training Fast Simulators: *produce events at correct relative proportions*

At parton level, events should follow Matrix Element proportions

$$\sigma(x,\theta) = \sum_i |\mathscr{M}_i(x,\theta)|^2$$

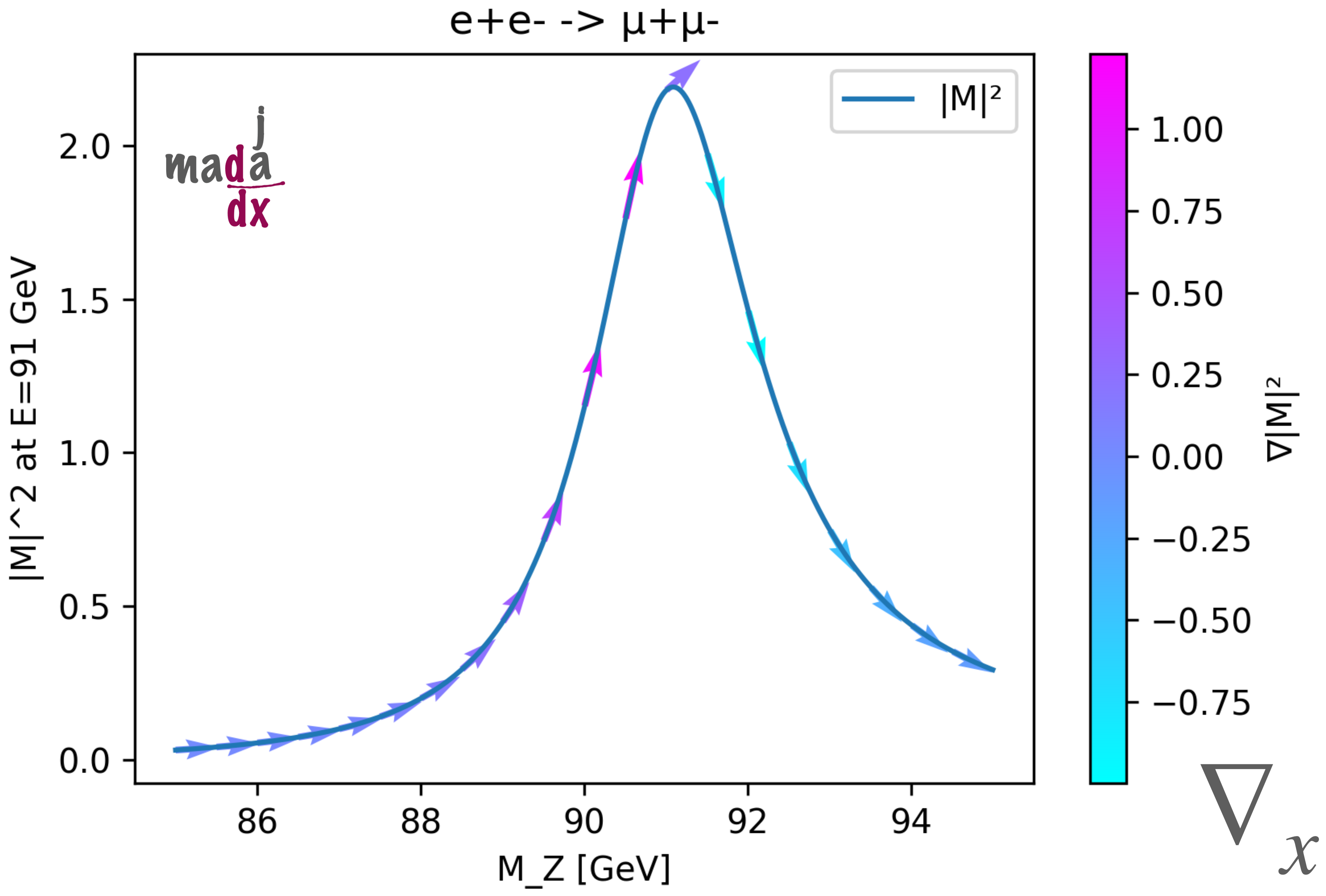


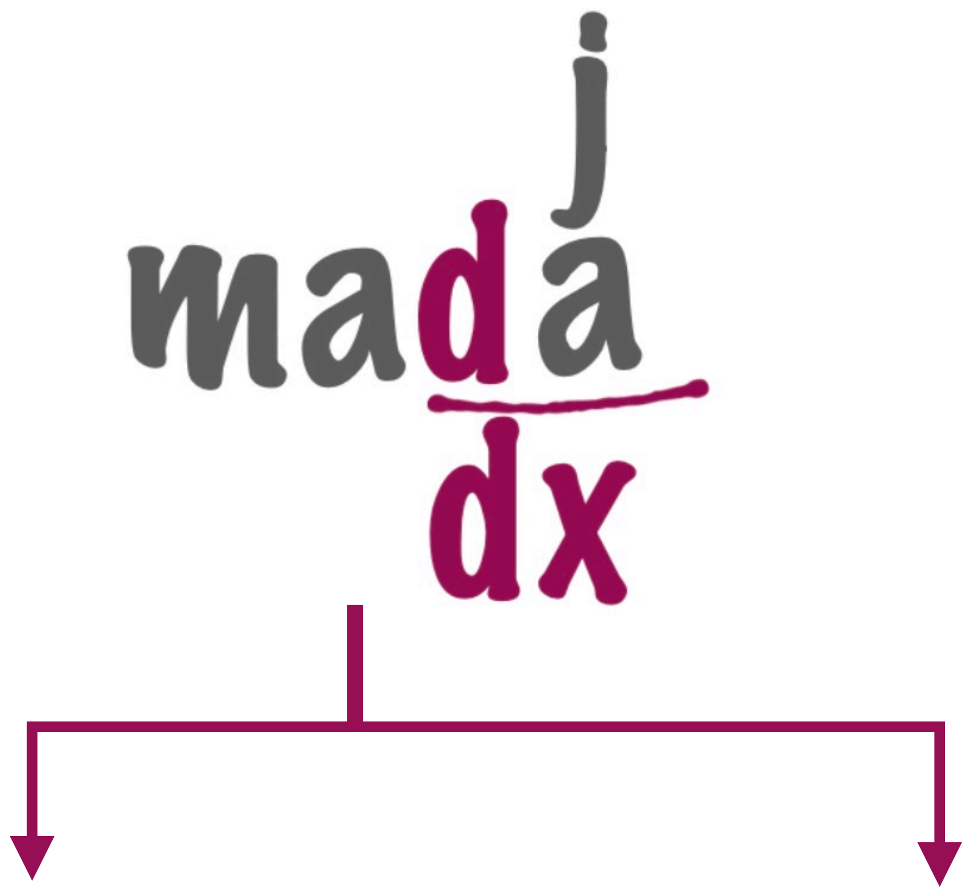
If we have ***differentiable Matrix Elements*** $|\mathscr{M}|^2(\{\vec{p_i}\},\theta)$ we can check directly



Differentiable Programming in ML

MadJax: MadGraph calculations (originally FORTRAN) transpiled into differentiable programming language (JAX) → **usable as evaluation function during training**



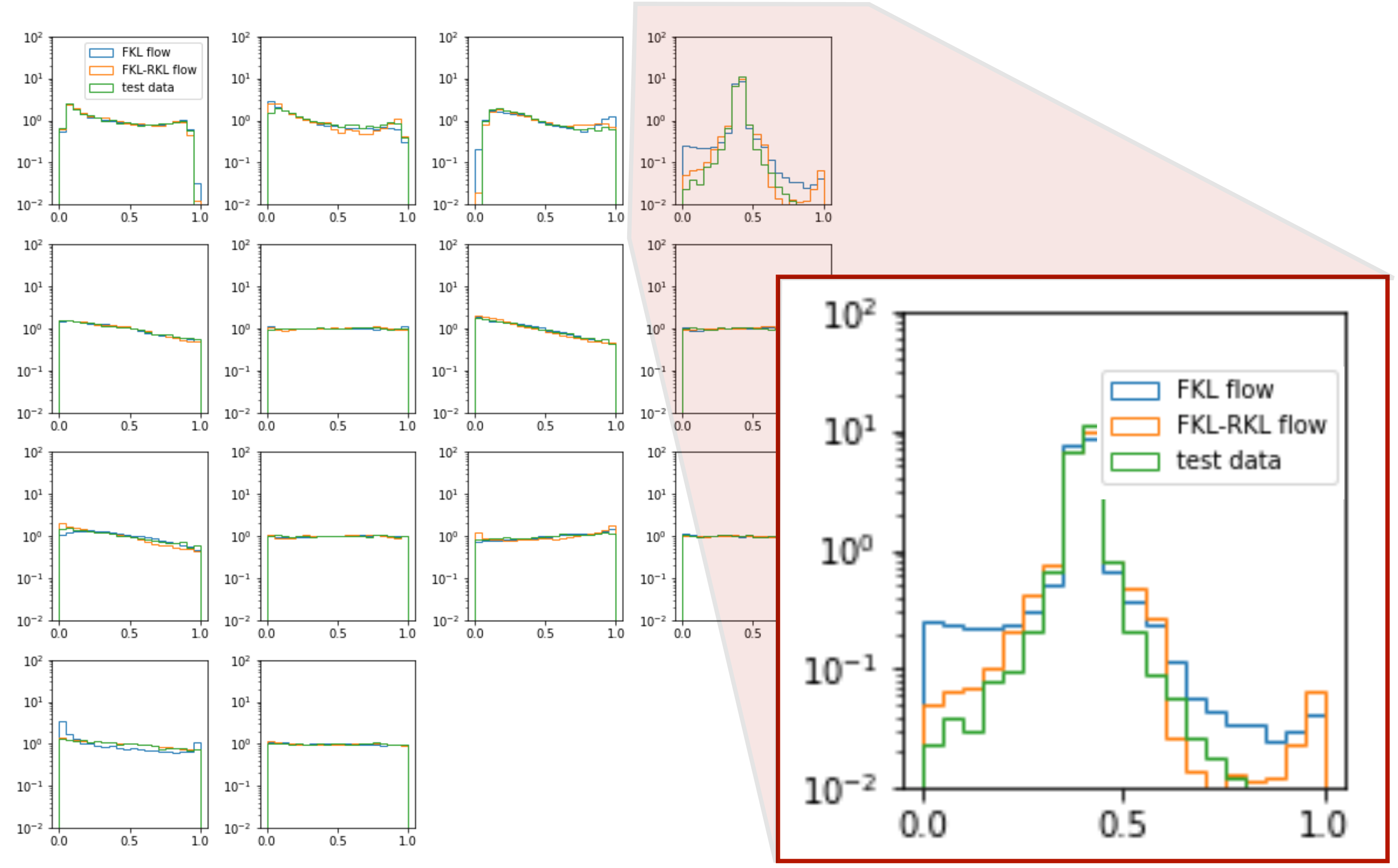


$$\nabla_x \sigma(x, \theta)$$

phase-space derivatives

$$\nabla_\theta \sigma(x, \theta)$$

theory Parameter derivatives



better description of density than pure ML training

```
mg5_aMC  —mode=madjax_me_gen -f ee_to_mumu.mg5
```

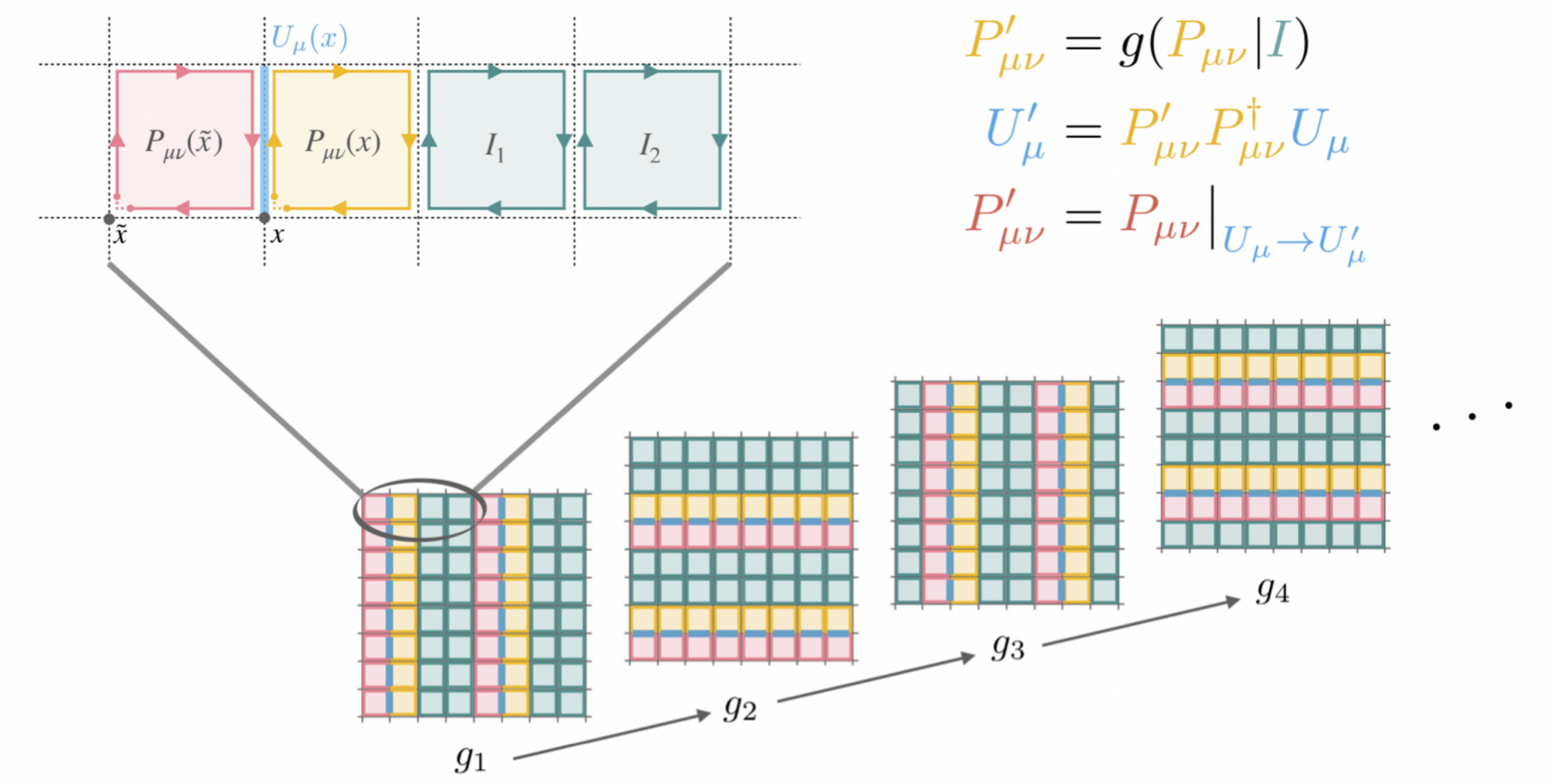
[LH, M. Kagan]
arxiv:2203.00057

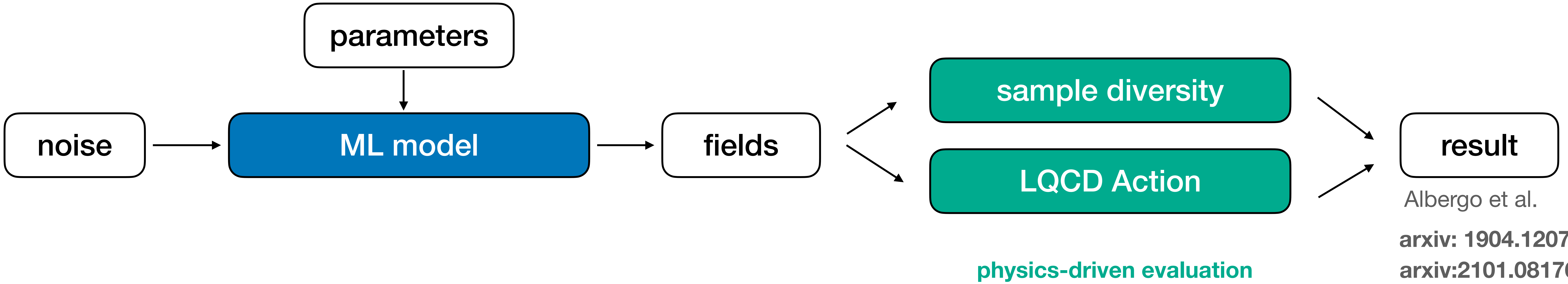
Differentiable Programming in ML

Same approach in Lattice QCD:

Learn **proposal distribution** for sampling of fields on a lattice (for MCMC / IS)

- encode symmetries in ML sampler
- evaluate on LQCD action in DiffProg language (pytorch)





Albergo et al.

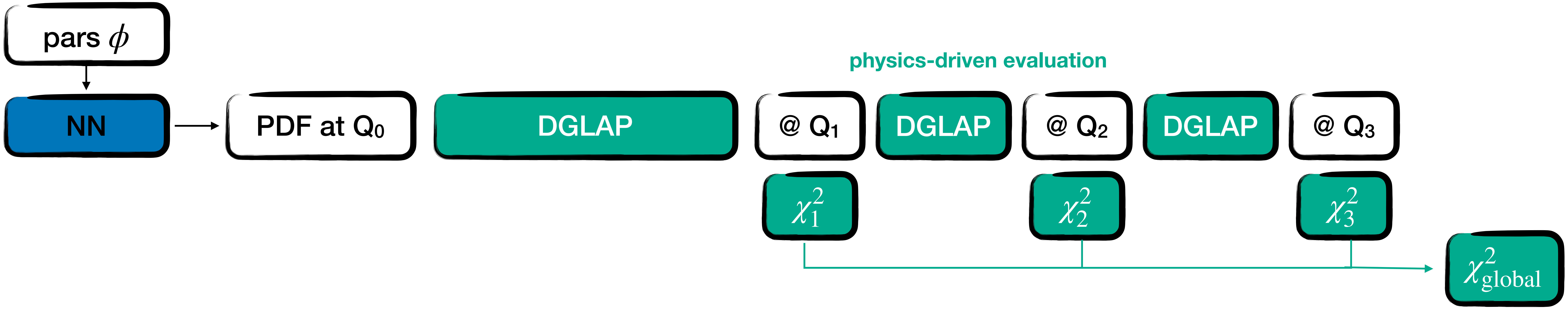
arxiv: 1904.12072

arxiv:2101.08176

Differentiable Programming in ML

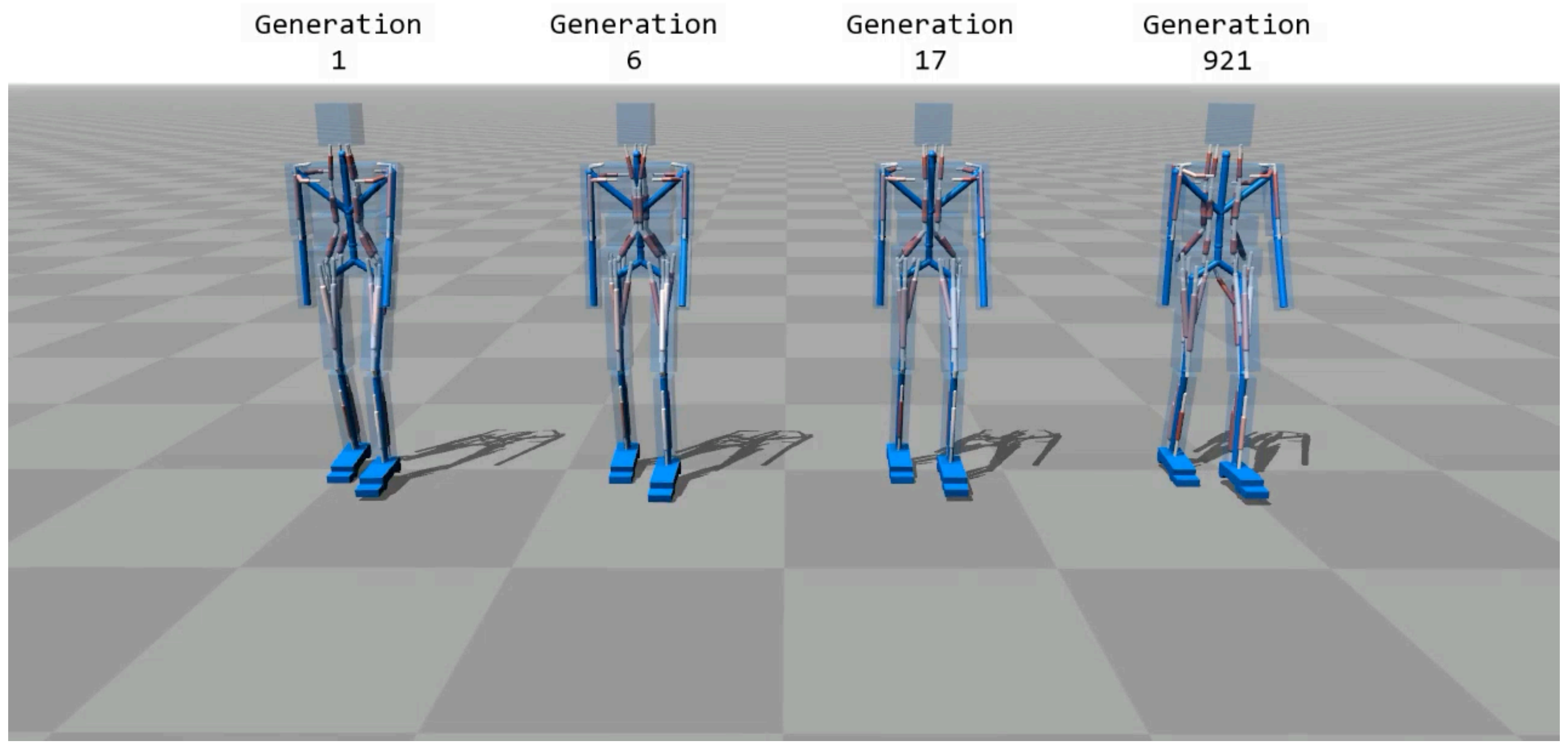
Parton Density Functions: DP can train NNPDF as it was meant to be trained

One of the early use-cases of NNs in HEP: PDF parametrizations



Curiosity:
traditionally **not(!) trained via gradient-descent**
→ too difficult to get gradients

→ use genetic algorithms (mutation + select)
→ works but is slow



genetic algorithms [Source]

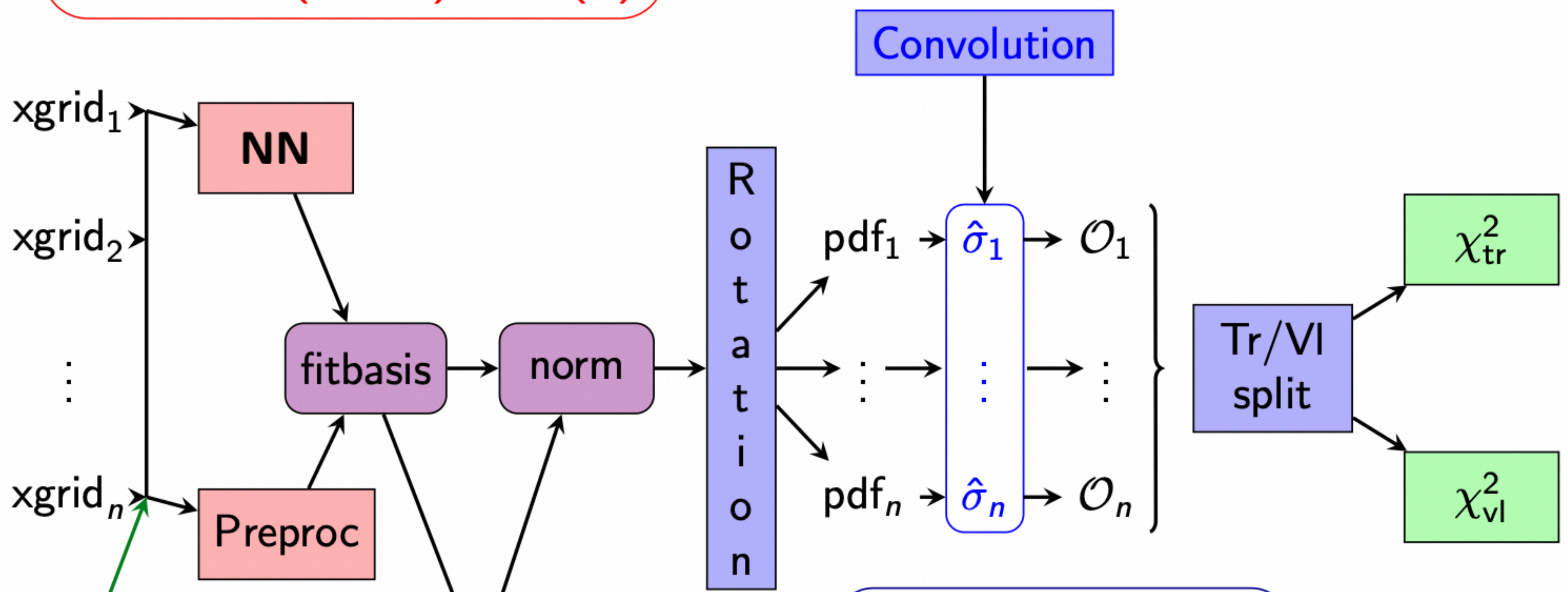
Differentiable Programming in ML

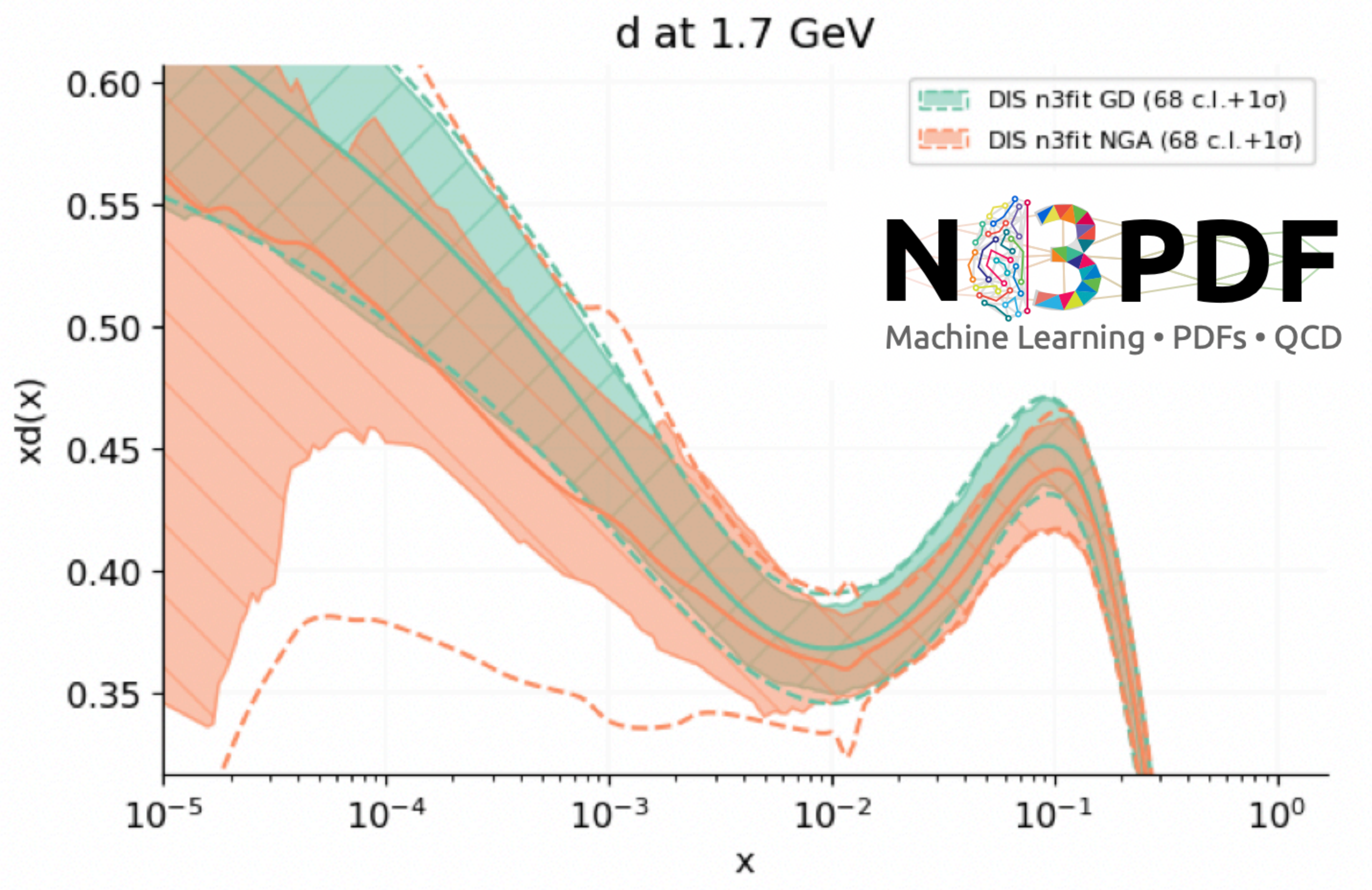
More recently: PDF evolution kernels implemented in DiffProg (Tensorflow)

- allows finally for a gradient-based training of NN

For all fits shown in this paper we utilize gradient descent (GD) methods to substitute the previously used genetic algorithm. This change can be shown to greatly reduce the computing cost of a fit while maintaining a very similar (and in occasions improved) χ^2-goodness. The less stochastic nature of GD methods also produces more stable fits than its GA counterparts. The main reason why the GD methods had not been tested before were due to the difficulty of computing the gradient of the loss function (mainly due to the convolution with the fastkernel tables) in a efficient way. This is one example on how the usage of new technologies can facilitate new studies thanks to differentiable programming and distributed computing.

$f_i = A_i x^{\alpha_i}(1-x)^{\beta_i} NN(x)$



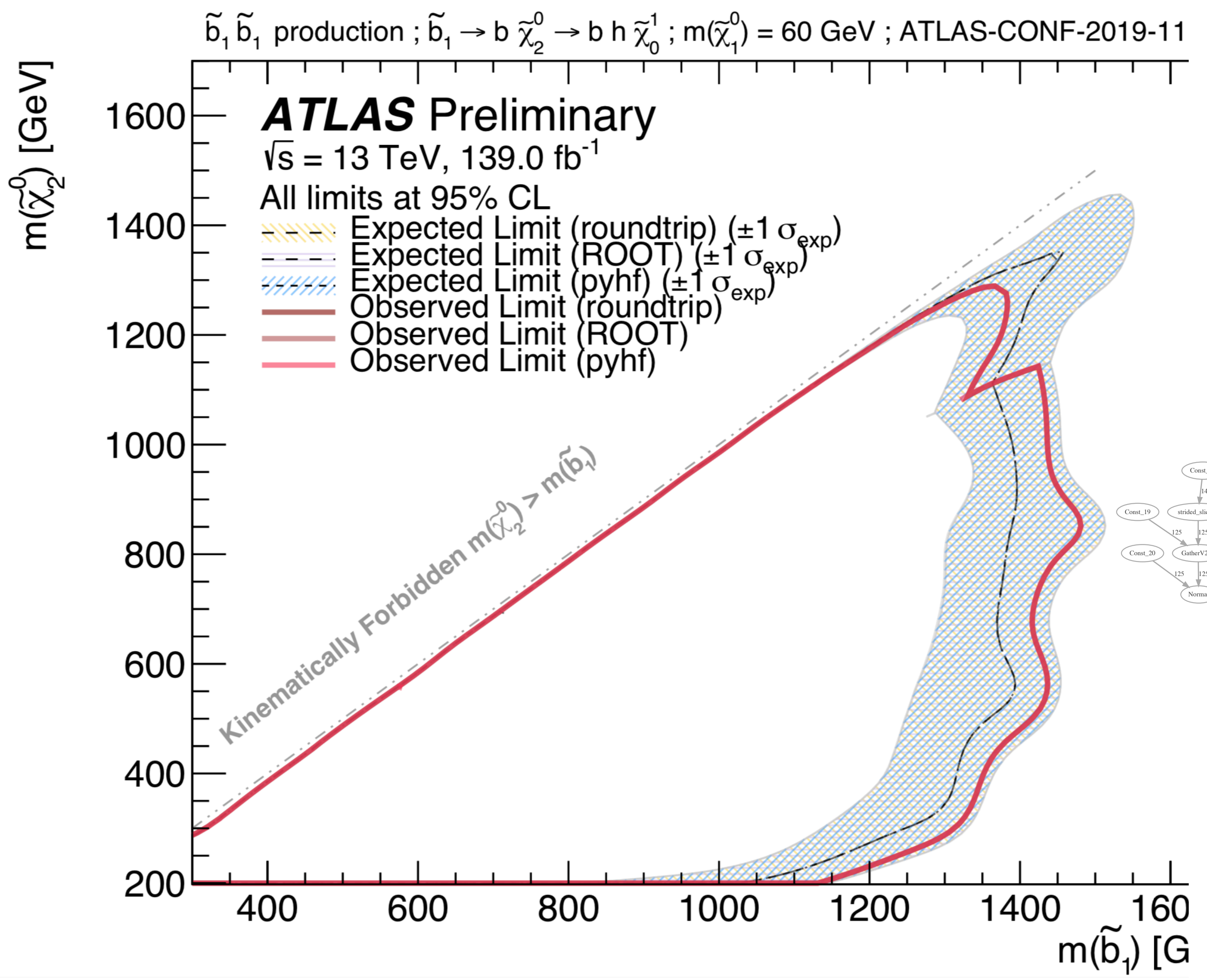


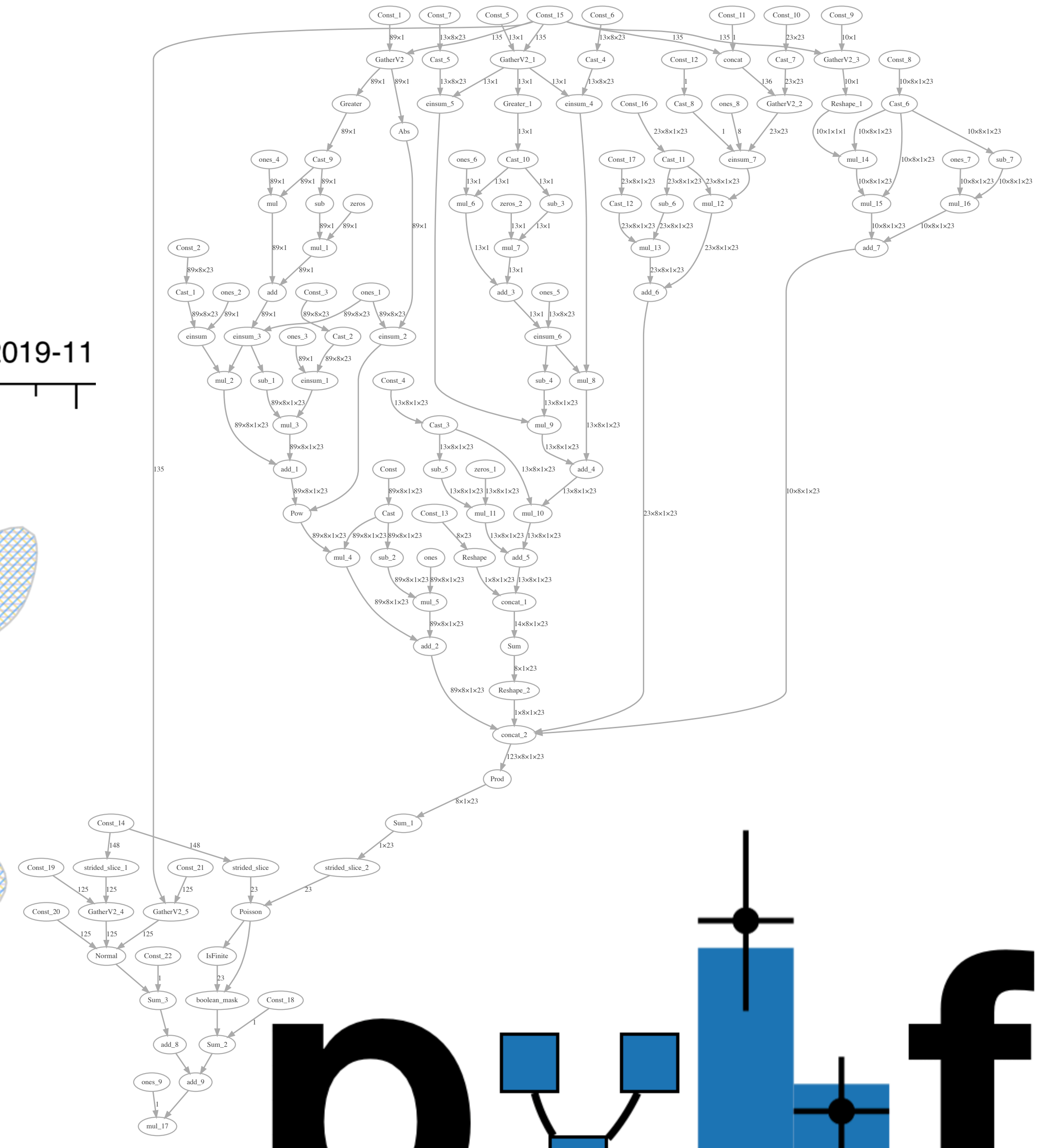
arxiv: 1907.05075
[Carrazza et al]

Differentiable Programming Beyond ML

Gradients useful far beyond ML: e.g. complex fits via differentiable programming

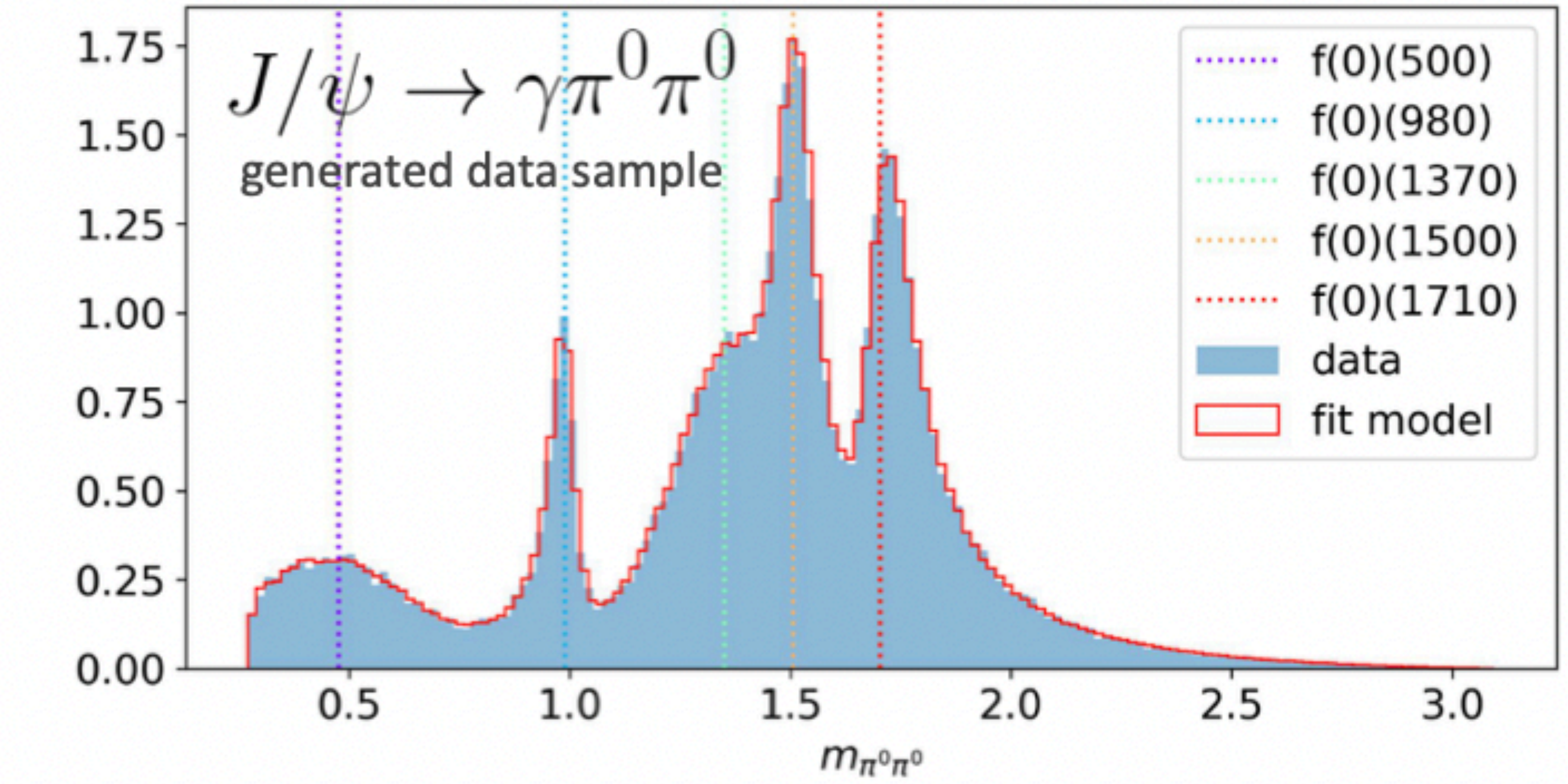
Binned Likelihoods (LHC, EIC, Belle-II, ...)





pyhf [LH, G. Start, M. Feickert]

Partial Wave Analysis




```
some_amplitude = model.components[
    R"A_{D^{0}_{0} \to K^{0}_{0} a_{0}(980)^{0}_{0}; a_{0}(980)^{0}_{0} \to K^{+}_{0} K^{-}_{0}}"
]
```

$$\frac{C_{D^0\to K^0_0 a_0(980)^0_0; a_0(980)^0 \to K^+_0 K^-_0}\Gamma_{a_0(980)^0} m_{a_0(980)^0}\sqrt{B_0^2\left(\left(d_{a_0(980)^0}\right)^2 q_{122}^2\left(m_{12}^2\right)\right)}D^0_{0,0}\left(-\phi_{1+2},\theta_{1+2},0\right)D^0_{0,0}\left(-\phi_{1,1+2},\theta_{1,1+2},0\right)}{-m_{12}^2+\left(m_{a_0(980)^0}\right)^2 - i m_{a_0(980)^0}\Gamma\left(m_{12}^2\right)}$$

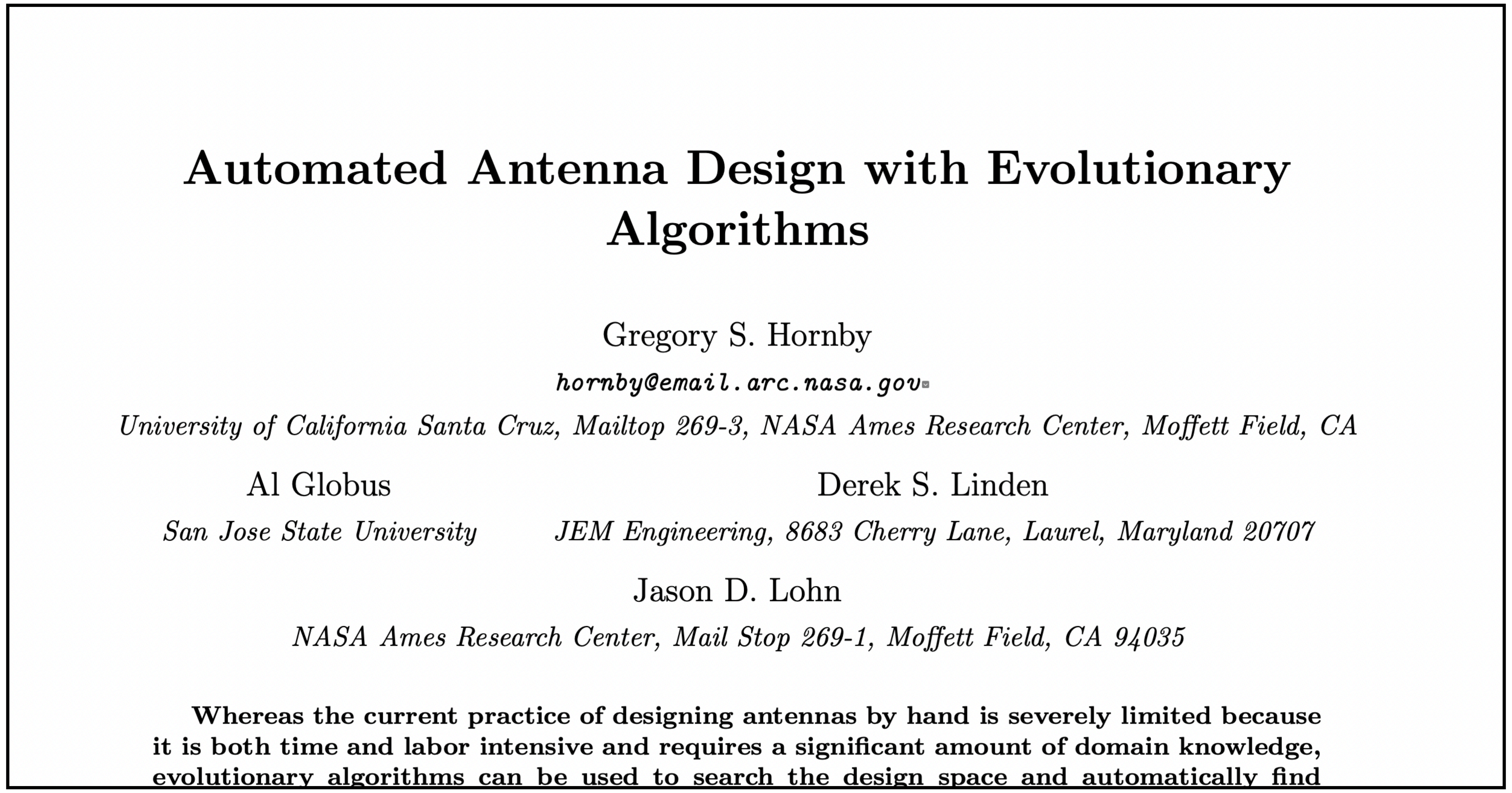
$$= \frac{C_{D^0\to K^0_0 a_0(980)^0_0; a_0(980)^0 \to K^+_0 K^-_0}\Gamma_{a_0(980)^0} m_{a_0(980)^0}}{-\frac{i\Gamma_{a_0(980)^0}\left(m_{a_0(980)^0}\right)^2\sqrt{\frac{\left(m_{12}^2-(m_1-m_2)^2\right)\left(m_{12}^2-(m_1+m_2)^2\right)}{m_{12}^2}}}{m_{12}\sqrt{\frac{\left(\left(m_{a_0(980)^0}\right)^2-(m_1-m_2)^2\right)\left(\left(m_{a_0(980)^0}\right)^2-(m_1+m_2)^2\right)}{\left(m_{a_0(980)^0}\right)^2}}} - m_{12}^2 + \left(m_{a_0(980)^0}\right)^2}$$

$$= \frac{0.074}{-m_{12}^2 + 0.96 - \frac{0.599\sqrt{m_{12}^2 - 0.975}}{m_{12}}}$$

ComPWA [R. deBoer, M. Mikhasenko]

Speaking of Genetic Algorithms…

Speaking of Genetic Algorithms…



Automated Antenna Design with Evolutionary Algorithms

Gregory S. Hornby
hornby@email.arc.nasa.gov
University of California Santa Cruz, Mailtop 269-3, NASA Ames Research Center, Moffett Field, CA

Al Globus
San Jose State University

Derek S. Linden
JEM Engineering, 8683 Cherry Lane, Laurel, Maryland 20707

Jason D. Lohn
NASA Ames Research Center, Mail Stop 269-1, Moffett Field, CA 94035

Whereas the current practice of designing antennas by hand is severely limited because it is both time and labor intensive and requires a significant amount of domain knowledge, evolutionary algorithms can be used to search the design space and automatically find

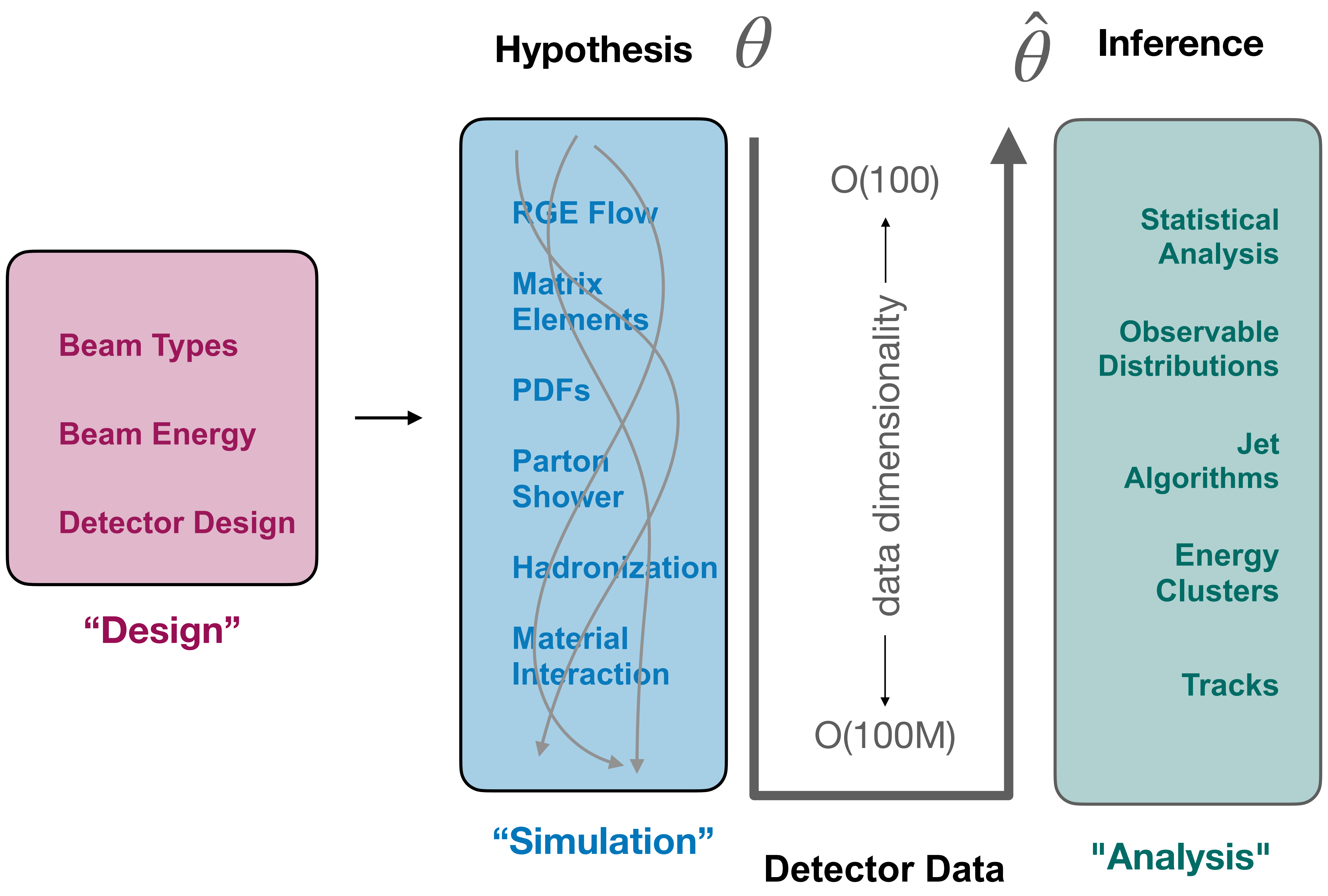
The current practice of designing and optimizing antennas by hand is limited in its ability to develop new and better antenna designs because it requires significant domain expertise and is both time and labor intensive.

Algorithmic Optimization of Hardware?

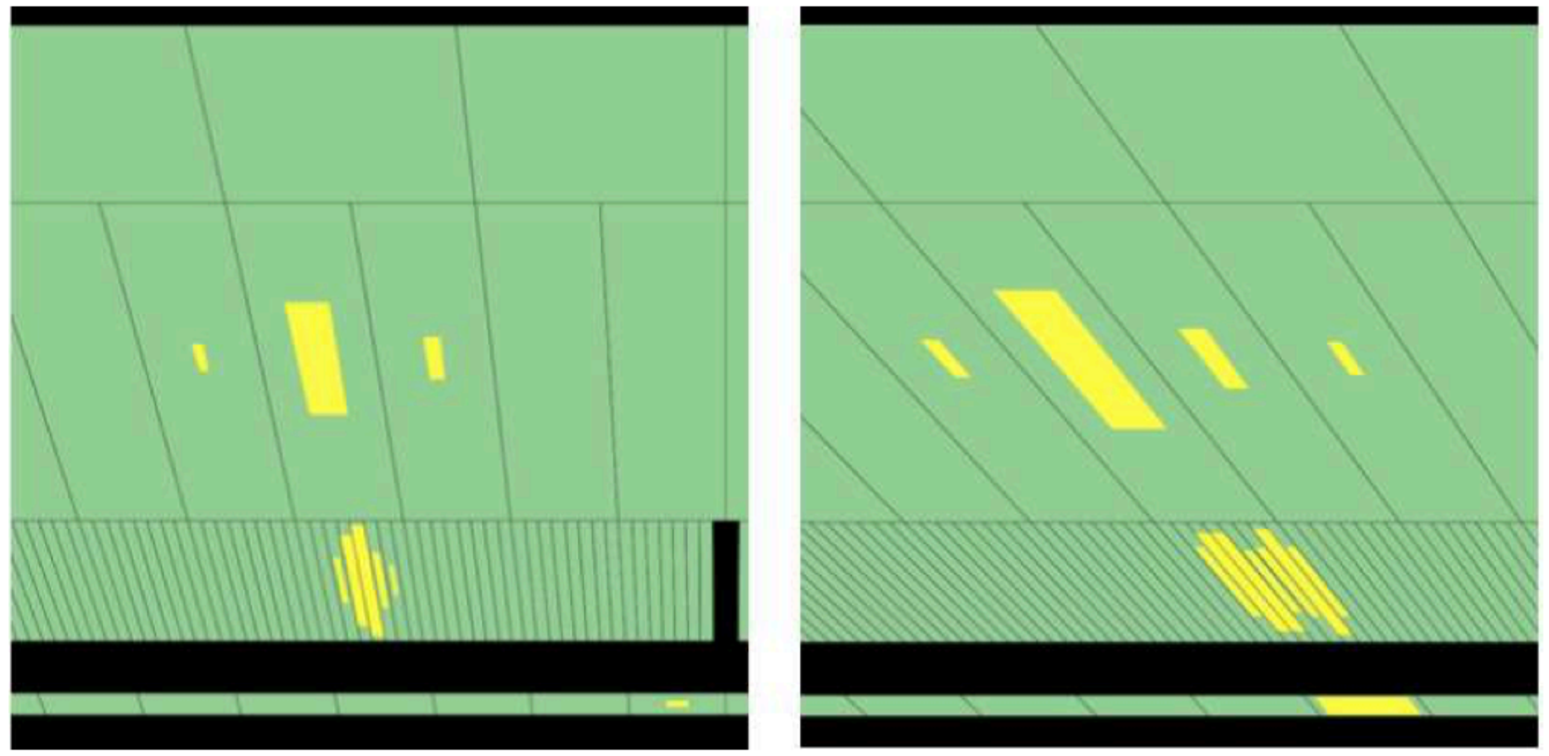
One more thing to tune:

Beyond reconstruction & analysis, we have an additional knob we can tune:

- the experiment design itself!



Example: ATLAS Calorimeter hand-designed for Higgs Discovery (Photon Pointing)



excellent photon identification and jet background rejection, by exploiting its fine longitudinal segmentation, thereby improving the signal to background ratio. Further, the diphoton invariant mass, defined as

$$m_{\gamma\gamma} = \sqrt{2E_1E_2\,(1-\cos\theta)}$$

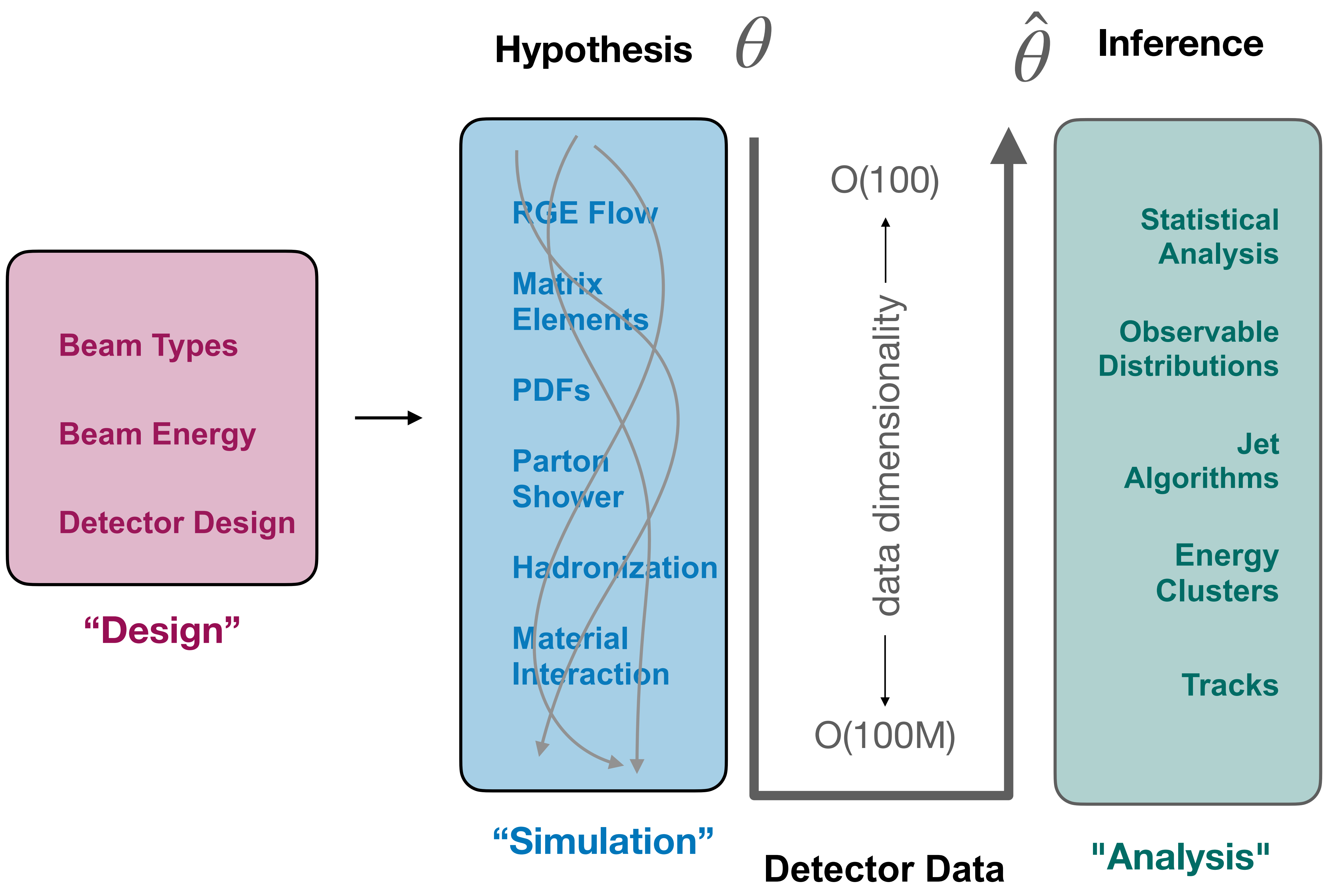
where E_1, E_2 are the two photon energies and θ is the angle

[Nikiforou, 1306.6756]

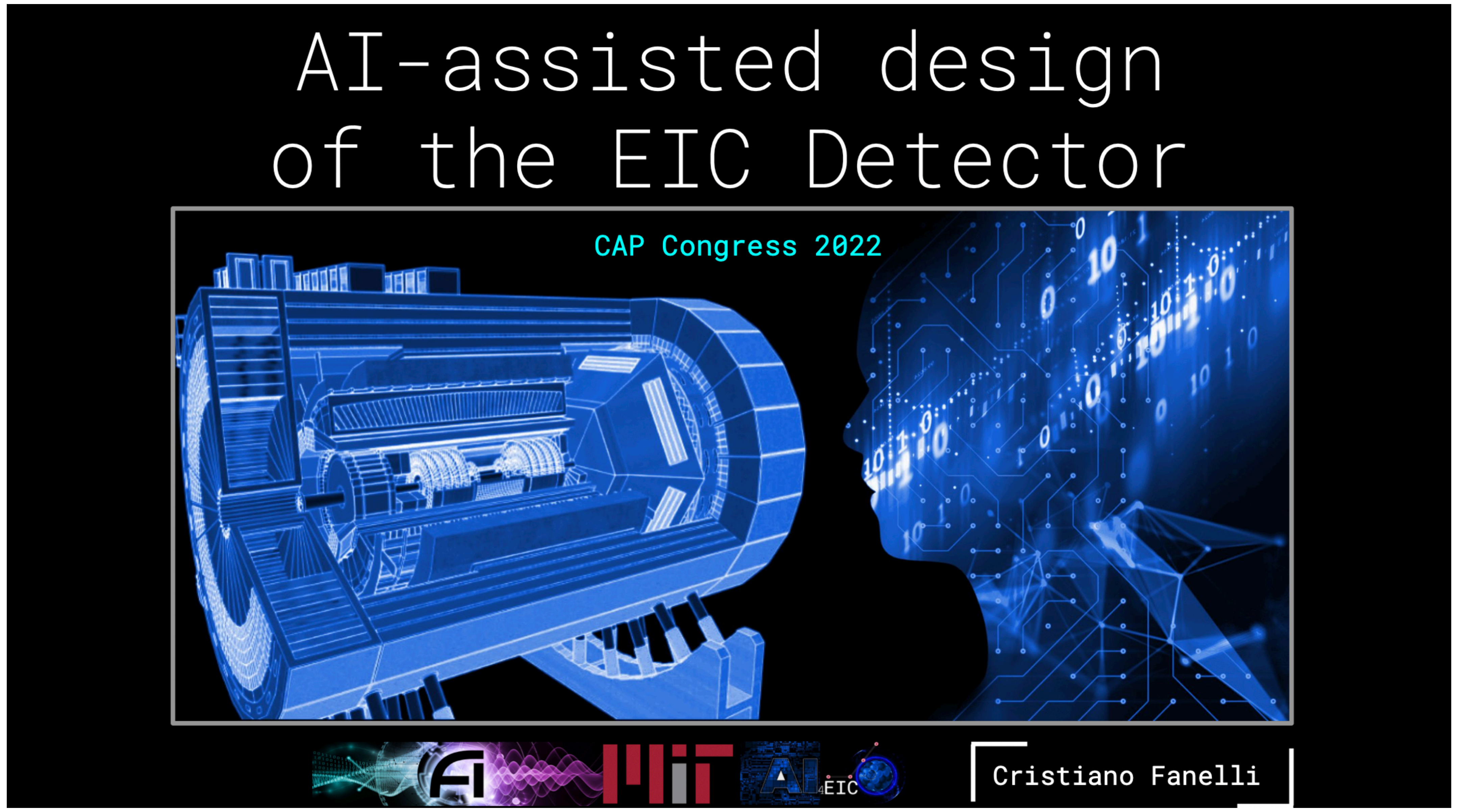
One more thing to tune:

Beyond reconstruction & analysis, we have an additional knob we can tune:

- the experiment design itself!



New Detectors are coming, can ML help design them?

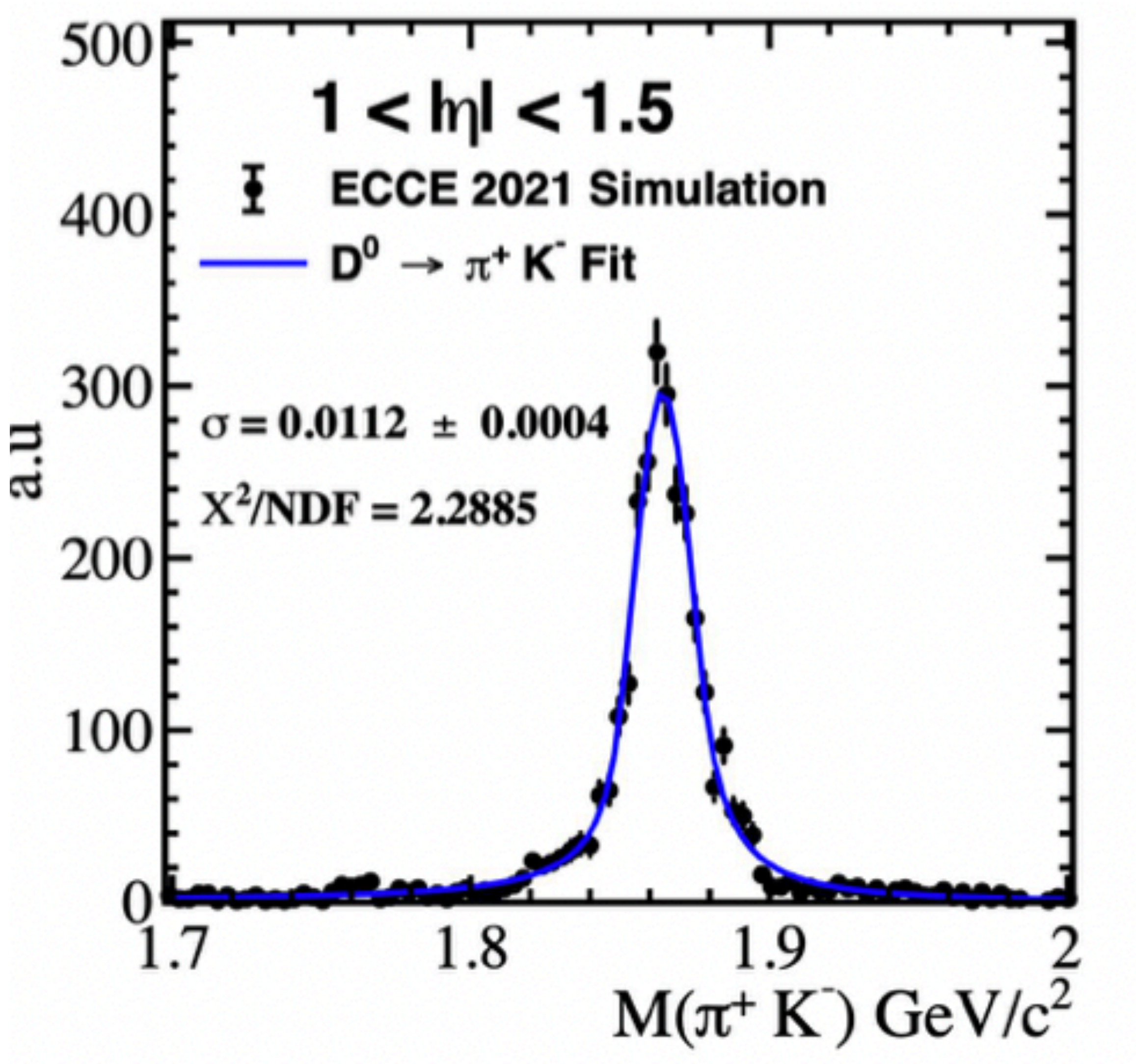


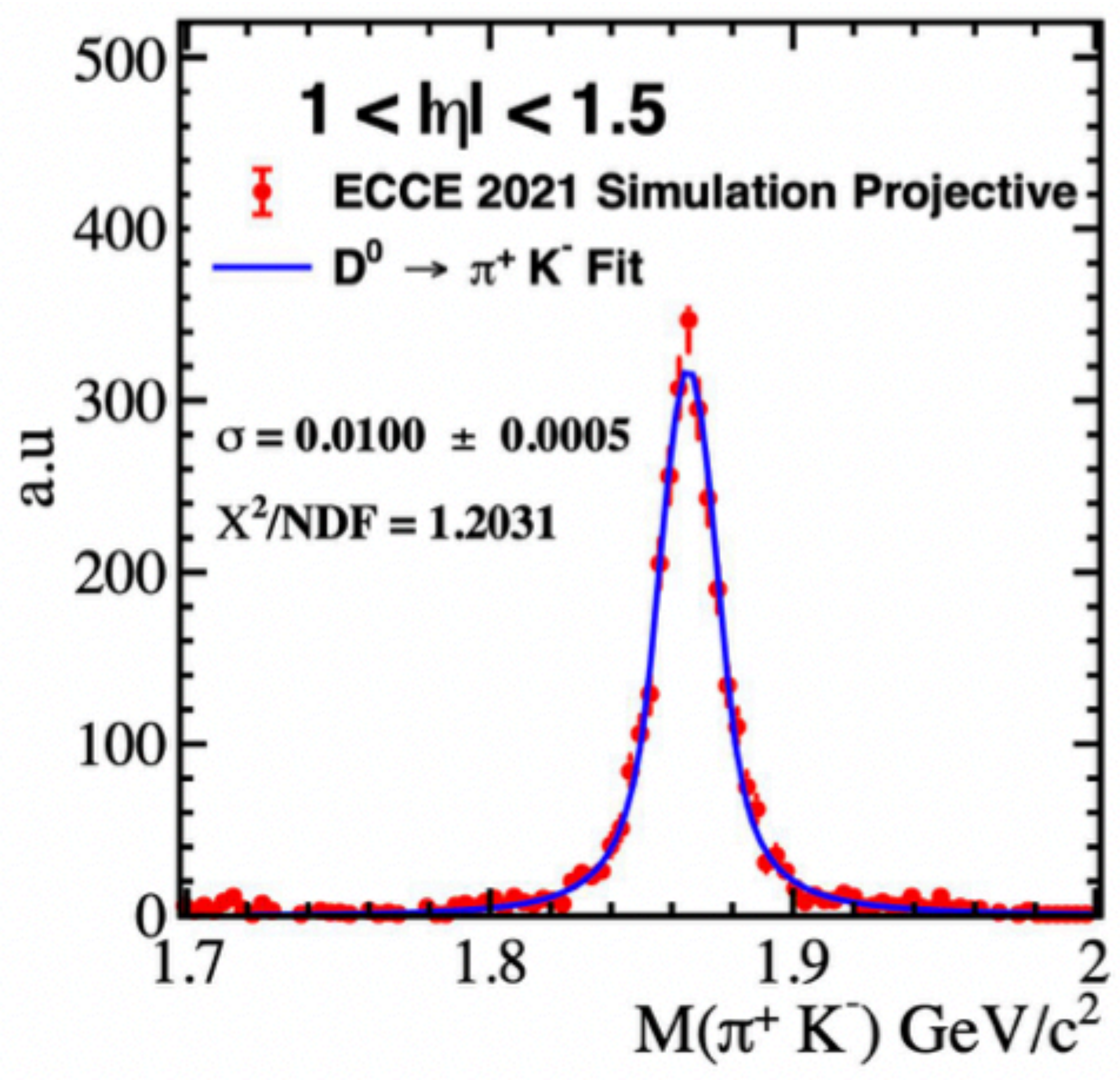
[AI4EIC]

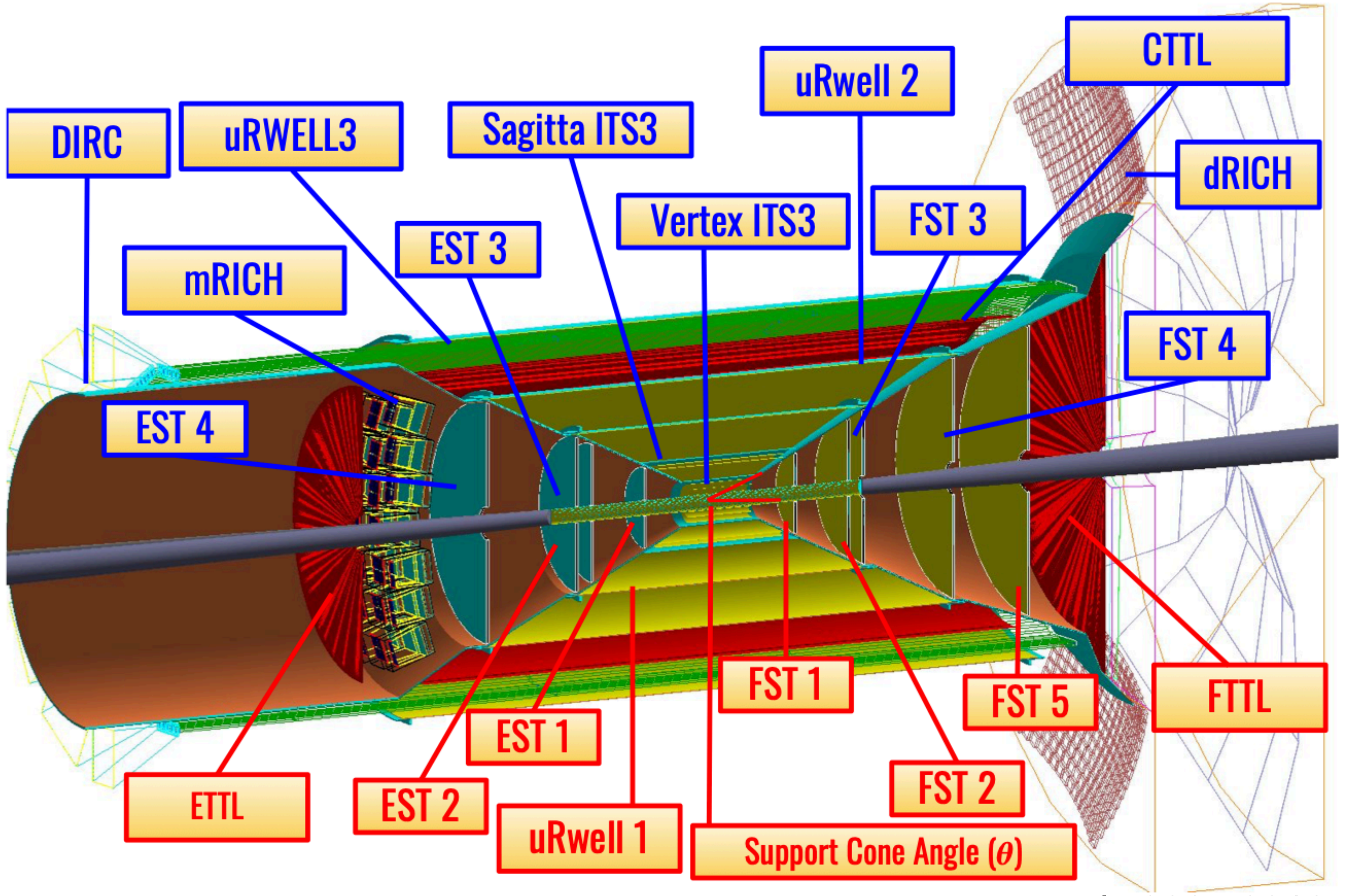
One more thing to tune:

Genetic Algorithms yield e.g. projective design of tracking system

→ ongoing R&D: 10% improvement in resolution



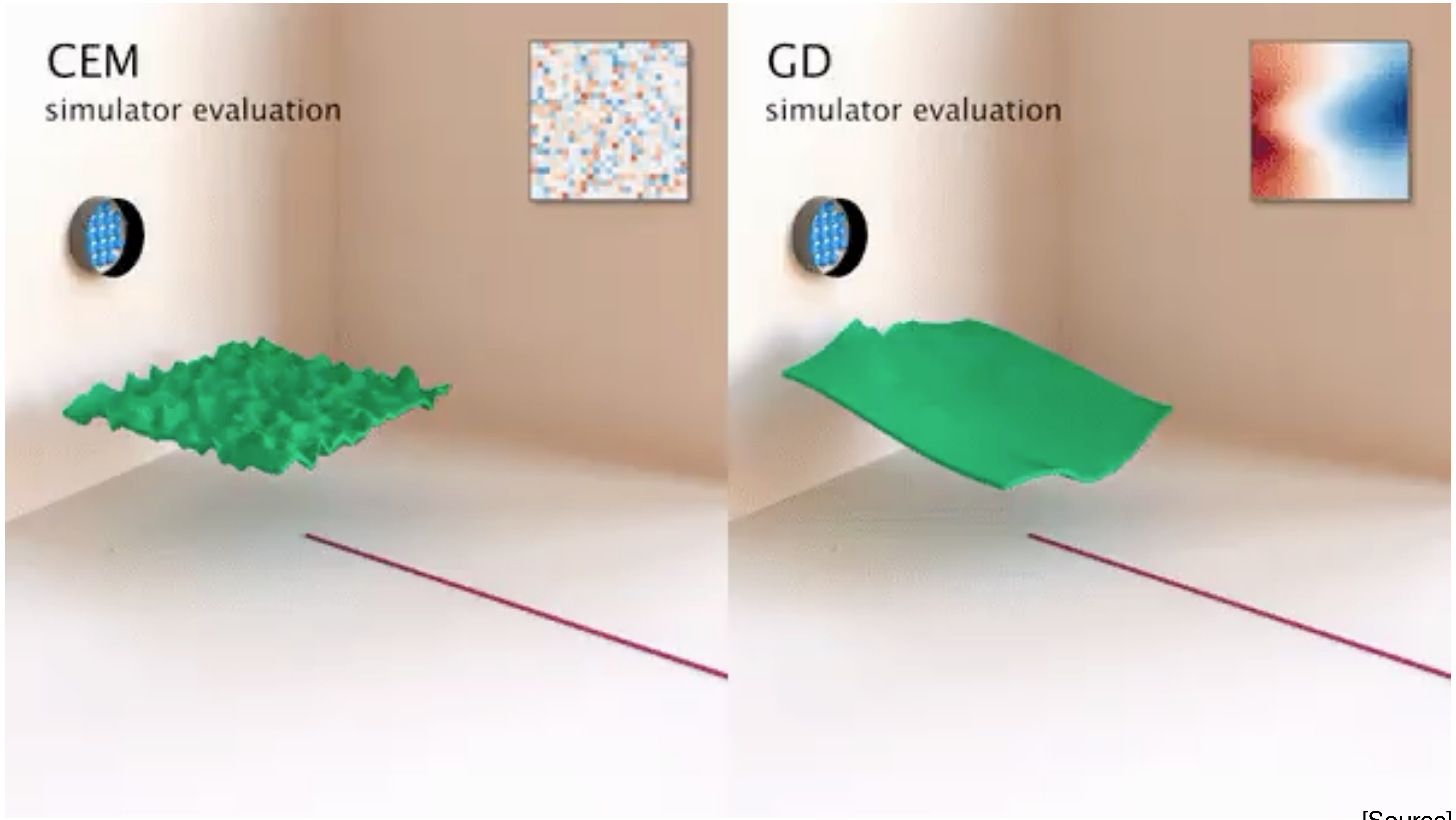




arxiv:2205.09185

Can a gradient-based optimization work / improve?
(similar to NNPDF example?)

An Example from outside Physics



[Source]

unoptimized design　　　　optimized design

param → Design → surface → Simulation → water flow → Evaluation → hit target?

Differentiable Design Optimization

Key difficulty: HEP simulation is highly stochastic and discrete (decays, showers,…)

→ need gradient over complex expectation over data and event histories

$$\nabla_\phi \mathbb{E}_x[f(x)]_\phi = \nabla_\phi \int \mathrm{d}x \mathbb{1}_{f(x)} \boxed{\int \mathrm{d}z\, p(x|z,\phi) p(z|\theta,\phi)}$$

gradient wrt
to fit paramets

expected performance

probability of data under design ϕ

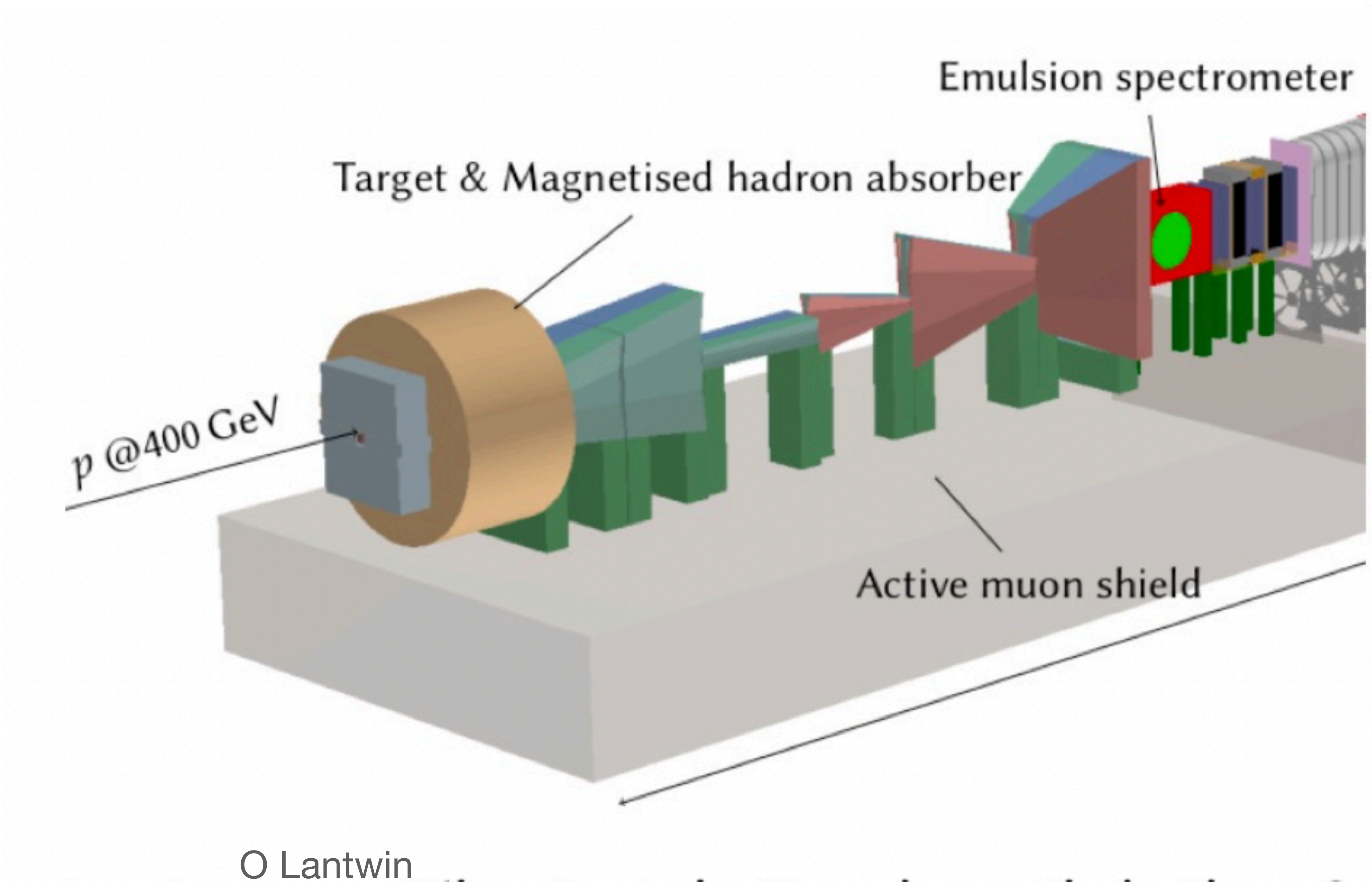
Ways to gradient-based training

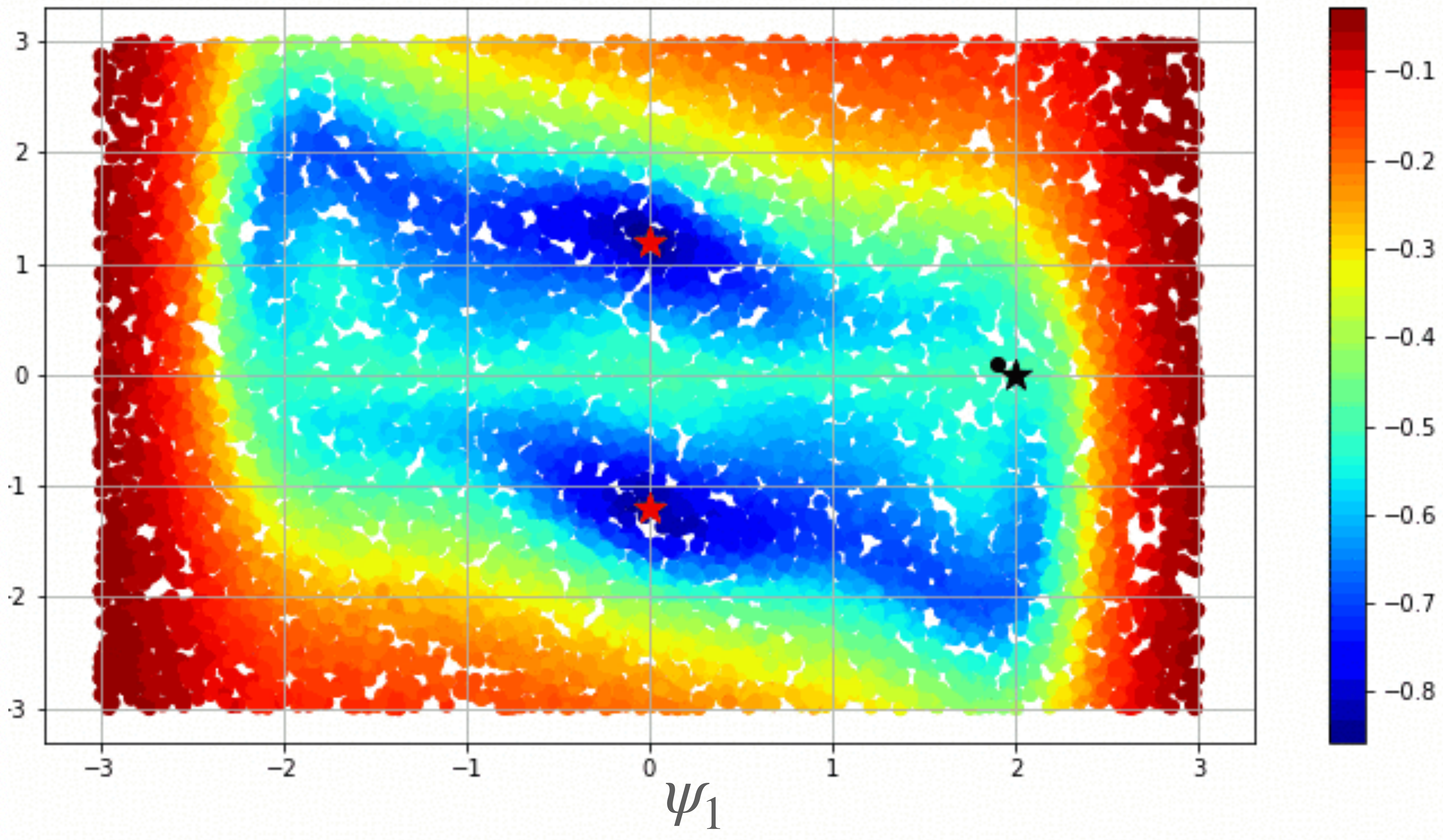
- replace true simulator with smooth, differentiable "surrogate" (ML generative model)
- neural network based proposal, train on gradients of policy instead of simulation

Differentiable Design Optimization

Example: Optimize Muon shielding in SHiP

- local differentiable proxy + gradient descent



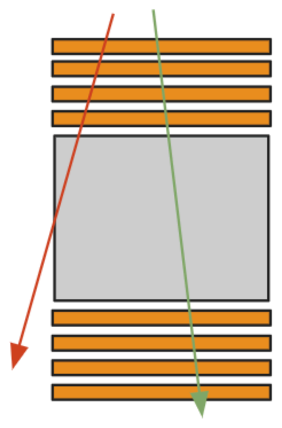


[see A.Ustyuzhanin's Talk in last year's QCHS]

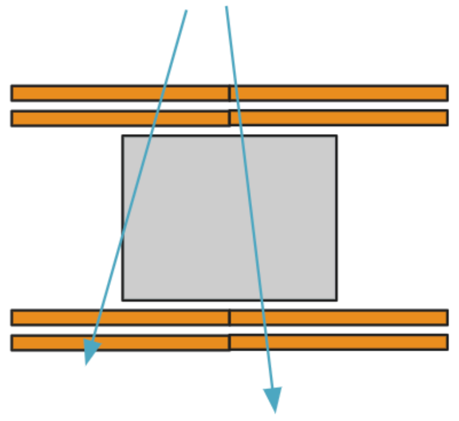
Differentiable Design Optimization

Example: End-to-end differentiable Muon Tomography Design Optimization

- instead of surrogate, implement a detector simulation in diffprog language
- fast convergence to good design

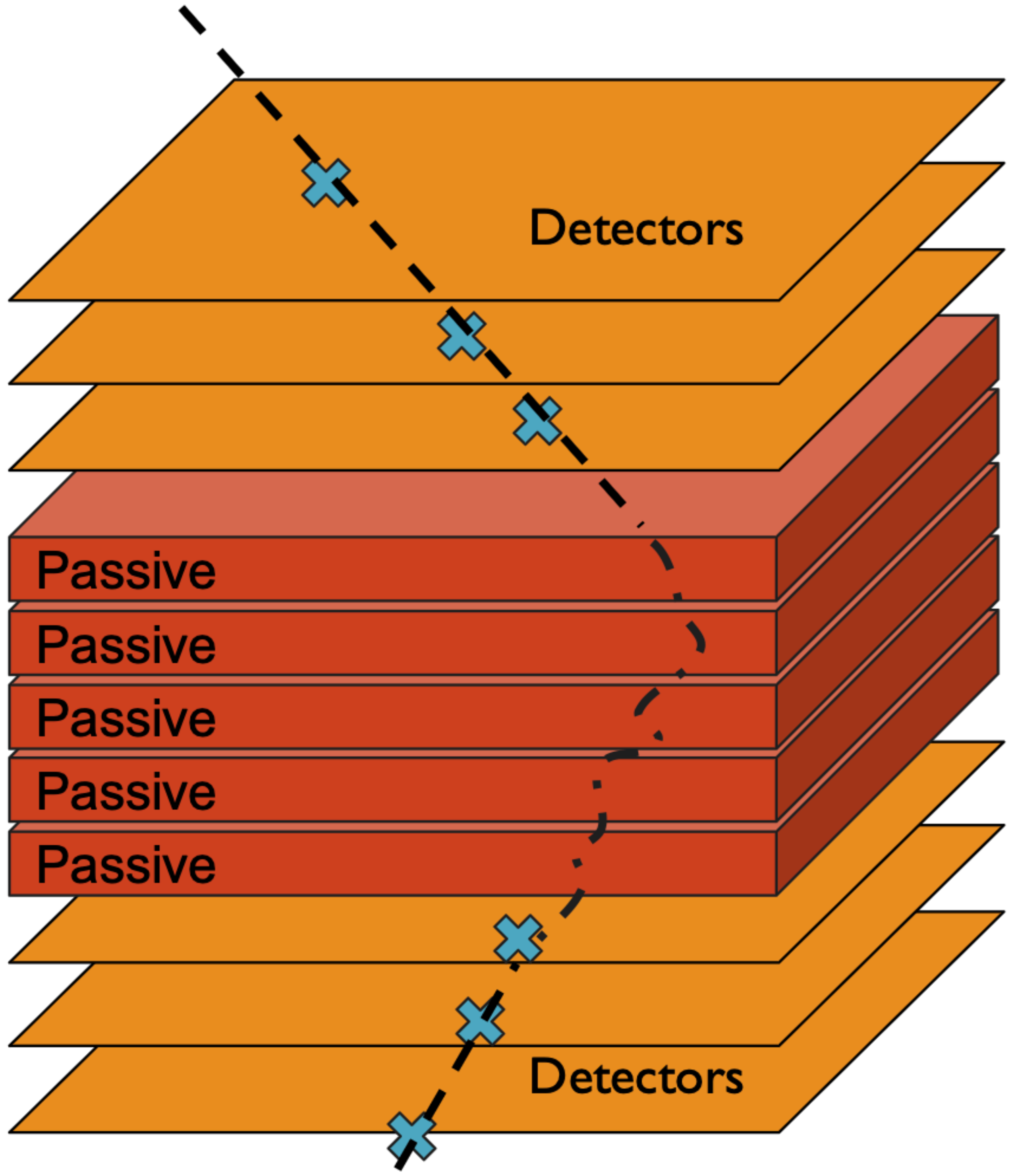


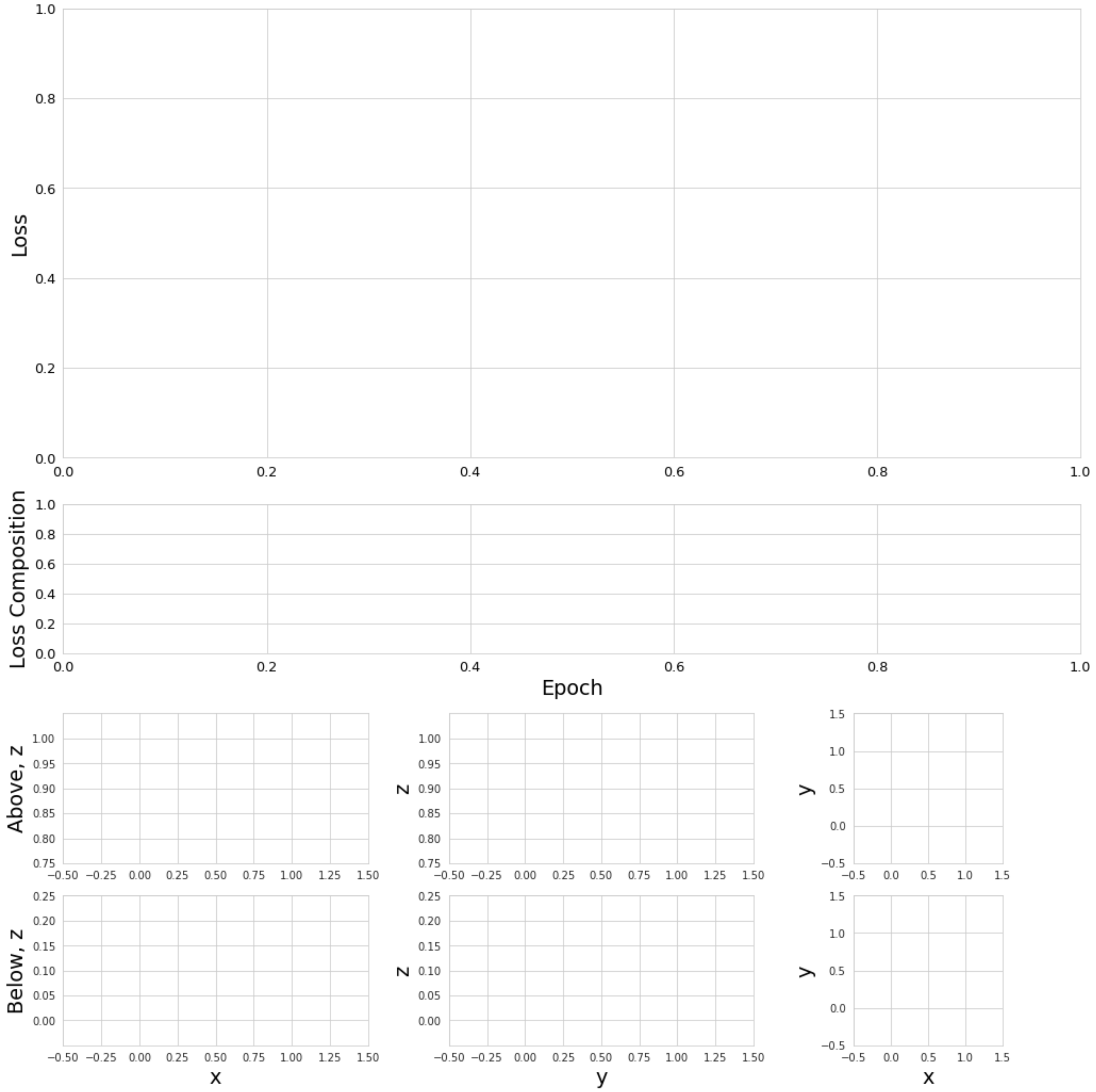
Example 1:
Muons
measured
precisely but
less efficiently



Example 2:
Muons
measured
less precisely
but more
efficiently

[G.Strong]





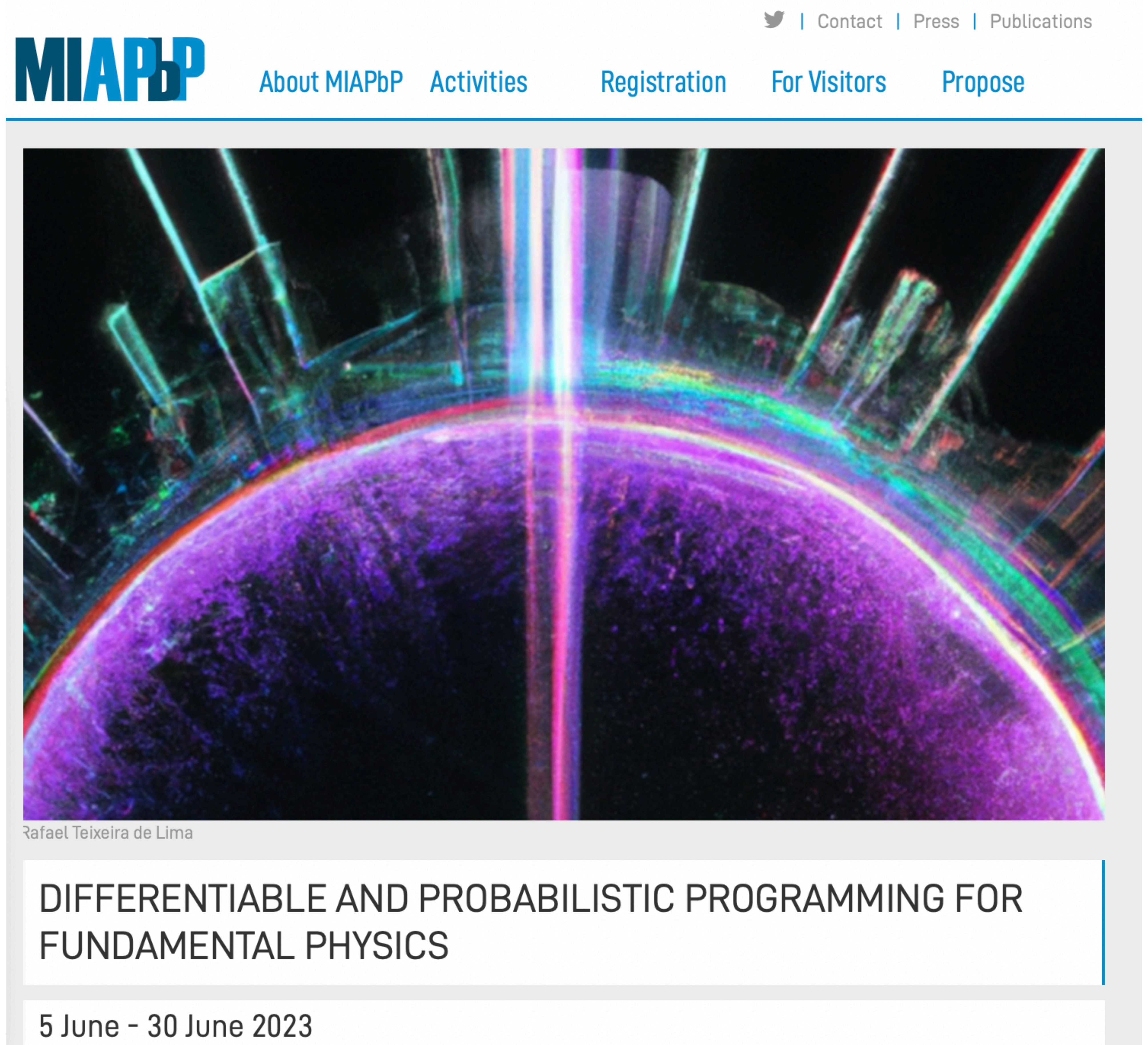
[see G. Strong's presentation Thursday Track H]

***Differentiable* Design Optimization**

Differentiable Simulators & Design Optimization: very early days

→ a lot of foundational work to be done - join!

https://indico.cern.ch/event/1145124/



https://www.munich-iapbp.de/probabilistic-programming

Summary

HNEP Analysis is dominated by simulation & optimization problems

- fast simulation, search for best observables
- ripe for significant improvement by ML methods

Differentiable Programming:

- one of the underlying secrets of Deep Learning, lots of interest in recent years
- allows a more nuanced look at ML: **encode physics** into model & evaluation

Generalizing from ML: we can abilities of diff. prog. to solve non-ML tasks

- nascent field of ML and/or gradient-based experimental design optimization e.g. for EIC