[img credit]

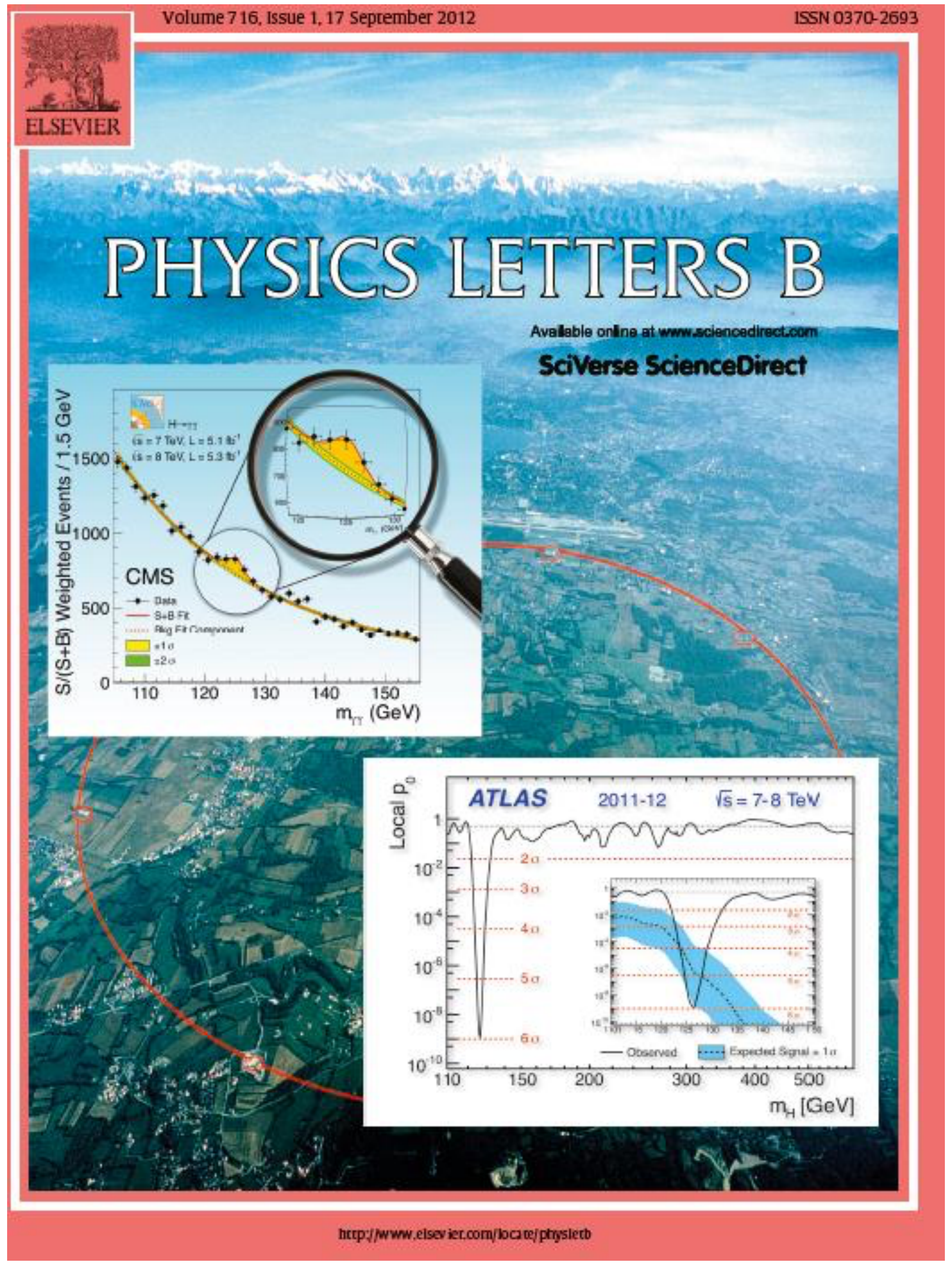
Differentiable Programming for High Energy Physics

Physics in LHC and Beyond - Matsue, Japan

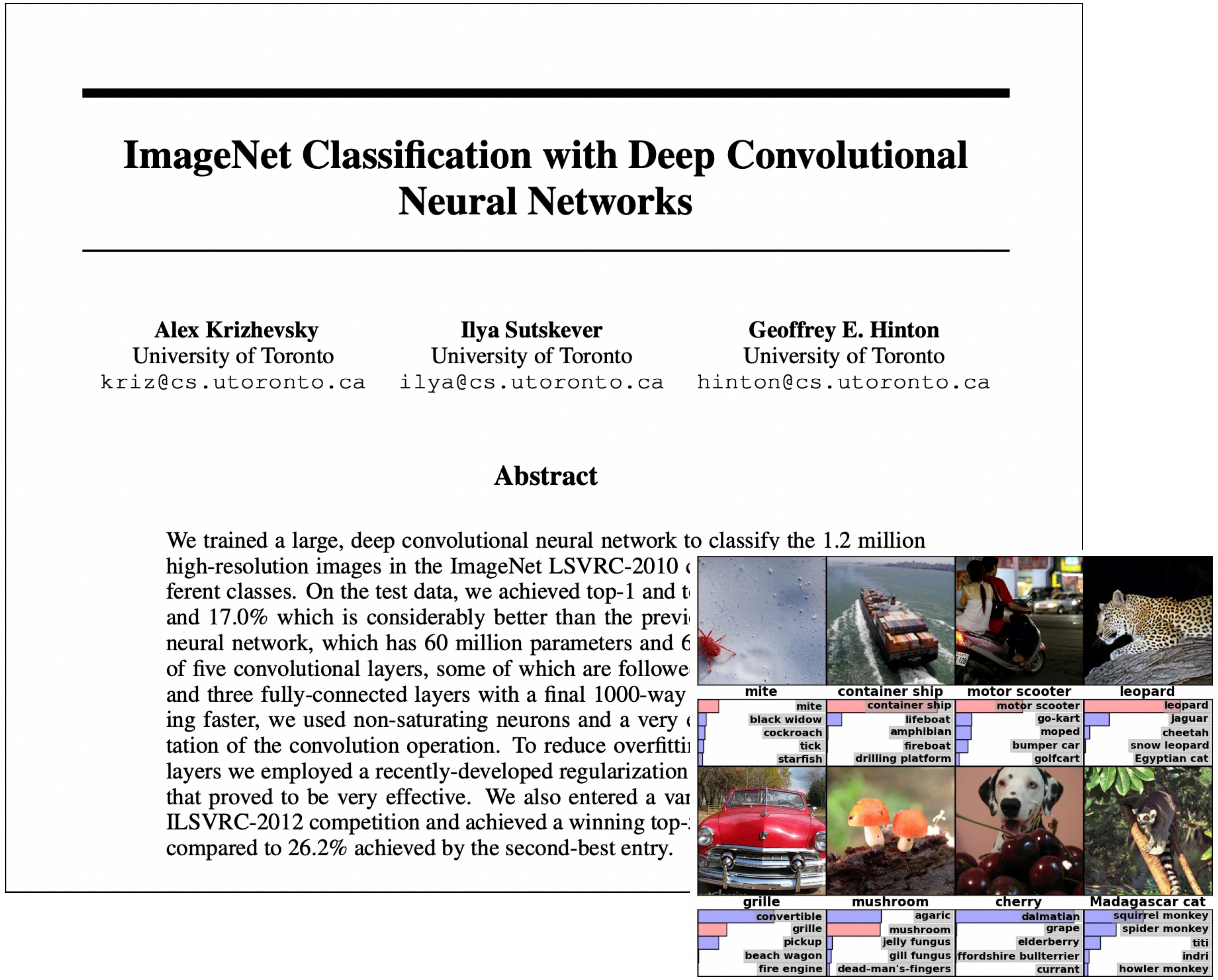
Lukas Heinrich, TUM

ODSL
ORIGINS Data Science Lab

Two Breakthroughs 10 Years ago



July 2012



ImageNet Classification with Deep Convolutional Neural Networks

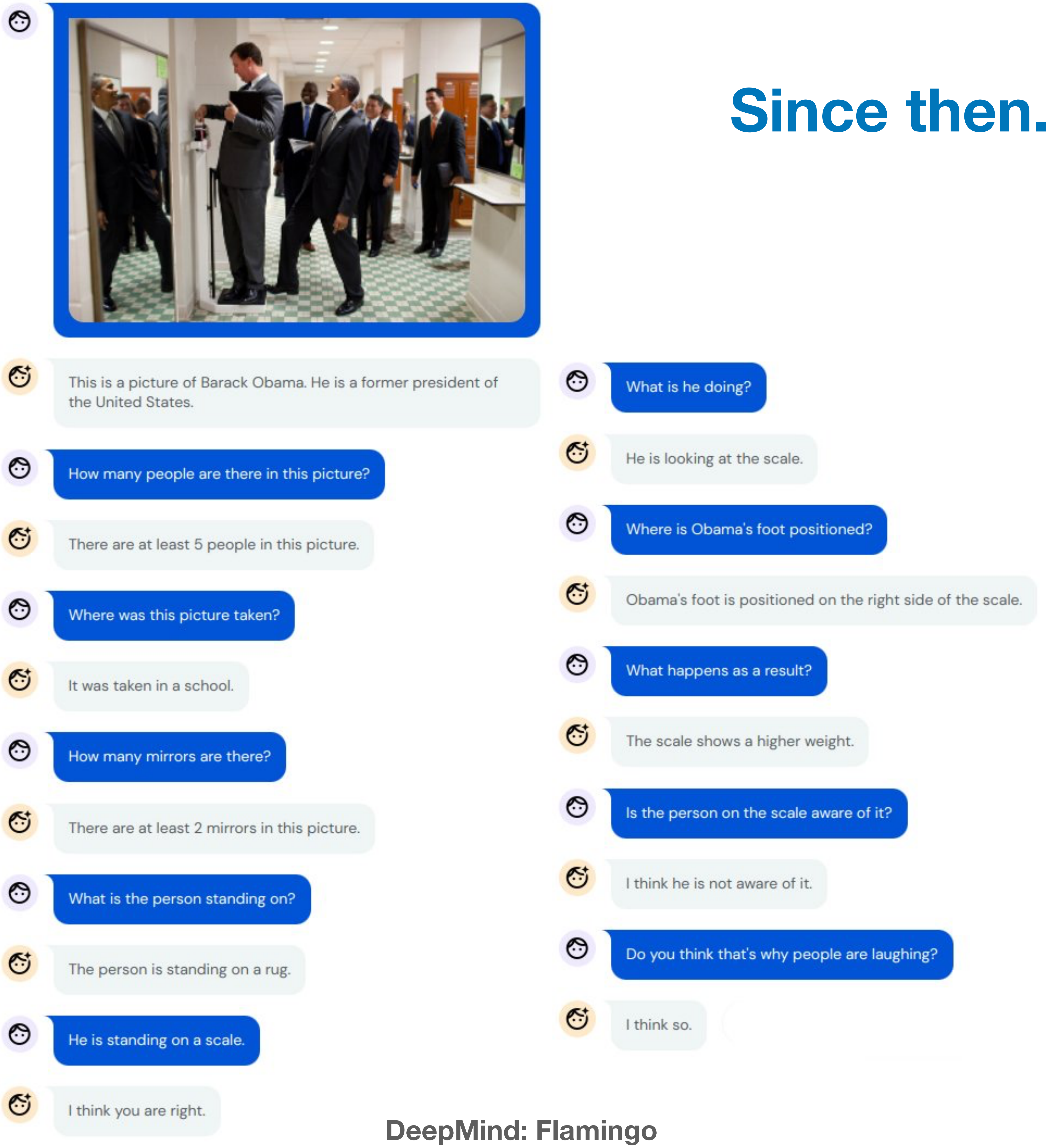
Alex Krizhevsky	**Ilya Sutskever**	**Geoffrey E. Hinton**
University of Toronto	University of Toronto	University of Toronto
kriz@cs.utoronto.ca	ilya@cs.utoronto.ca	hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 c ferent classes. On the test data, we achieved top-1 and t and 17.0% which is considerably better than the previ neural network, which has 60 million parameters and 6 of five convolutional layers, some of which are followe and three fully-connected layers with a final 1000-way ing faster, we used non-saturating neurons and a very e tation of the convolution operation. To reduce overfitti layers we employed a recently-developed regularization that proved to be very effective. We also entered a var ILSVRC-2012 competition and achieved a winning top- compared to 26.2% achieved by the second-best entry.

December 2012

Since then.. vast progress in ML



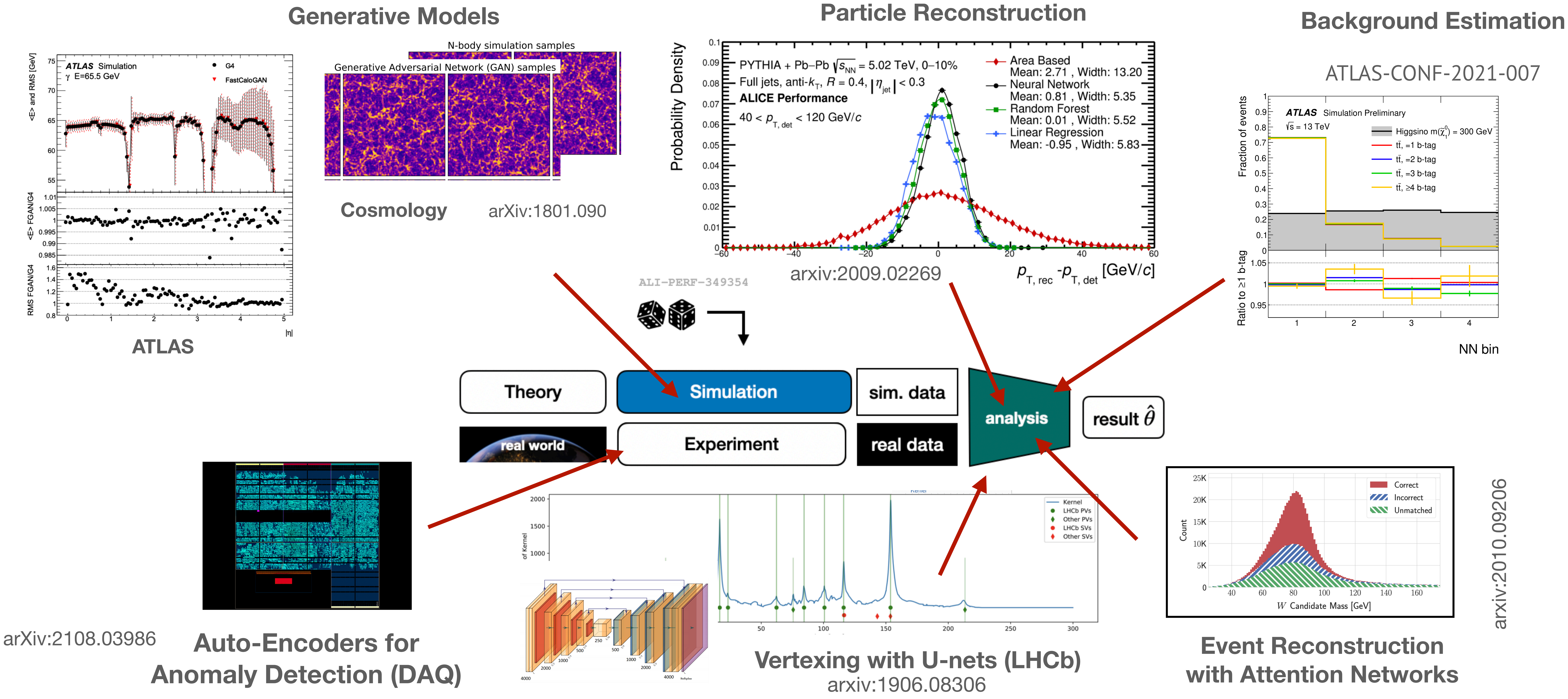
DeepMind: Flamingo

Prompt: "Panda Mad Scientist mixing sparkling chemicals, artstation"

↓

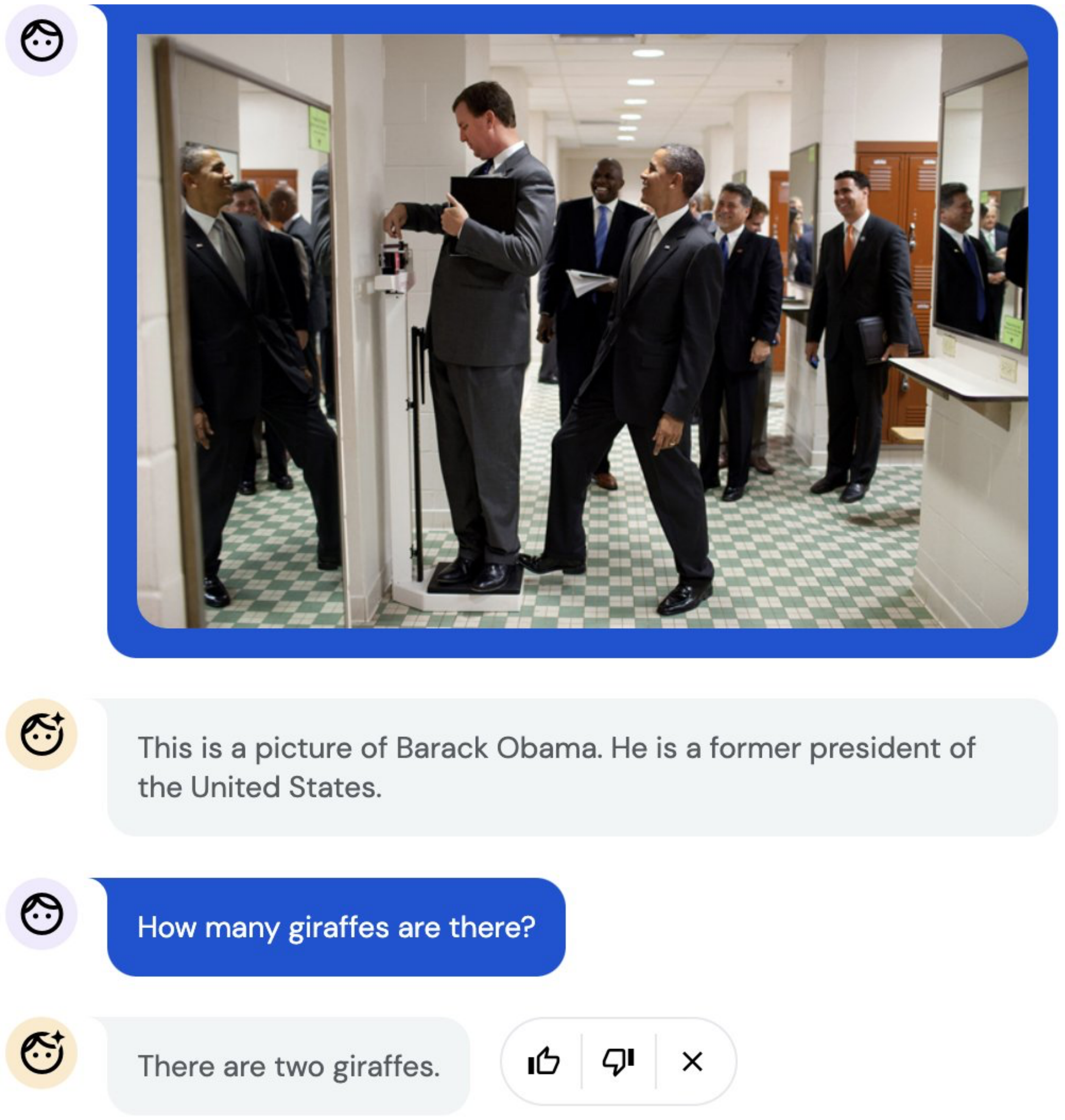
OpenAI: Dall-E 2

In HEP and Fundamental Science, it’s entering everywhere



Where is this going?

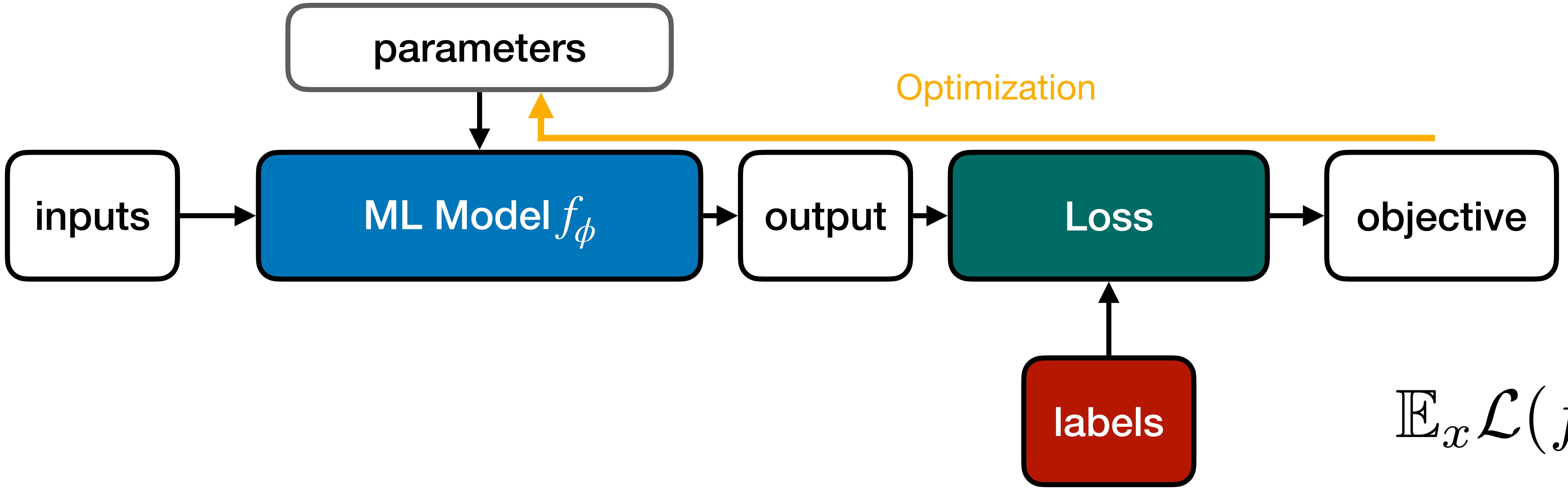
Challenges



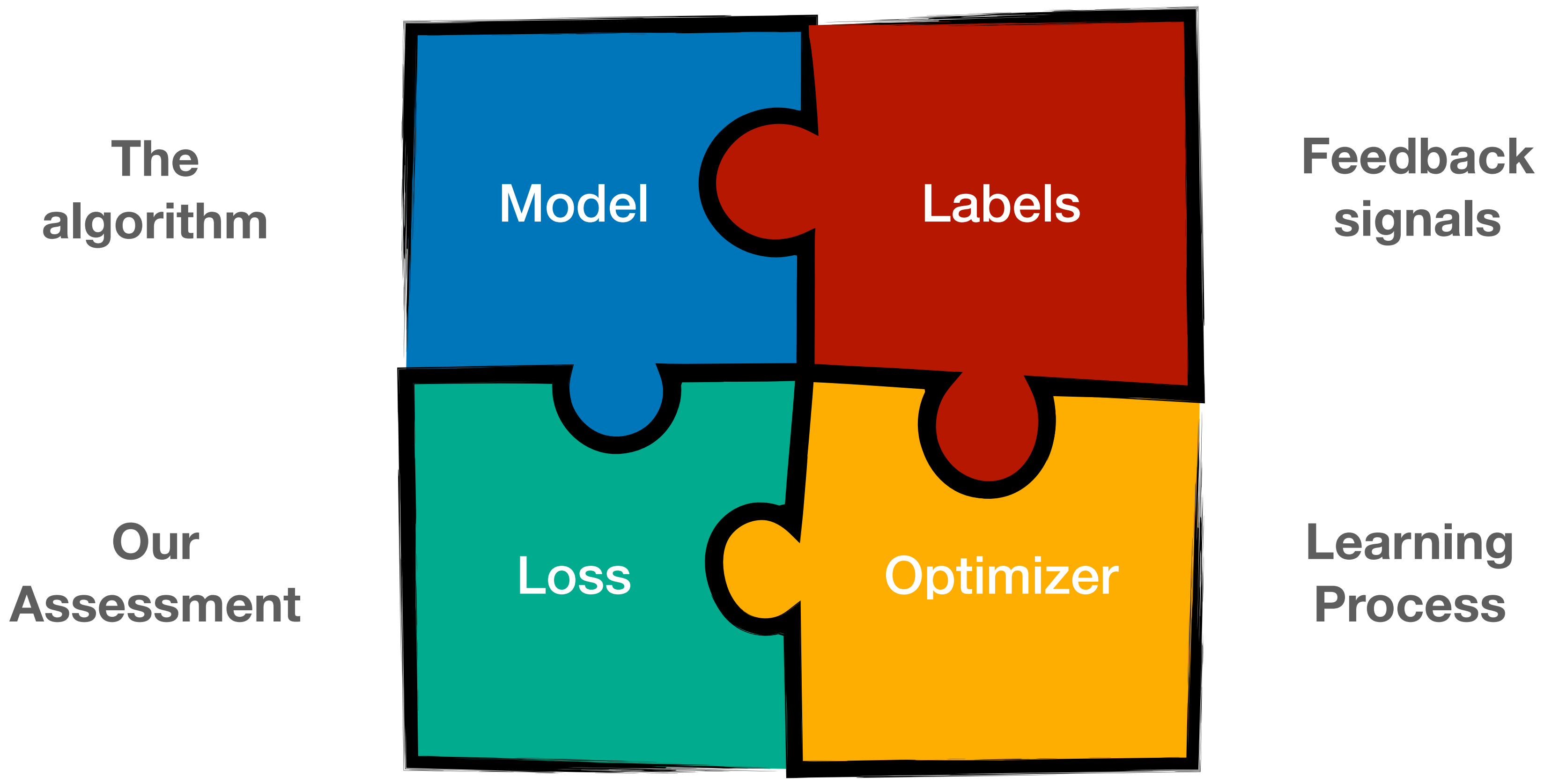
Replacing everything with one big black-box is not sufficient for scientific use cases: need uncertainties, interpretability, robustness, ….

A lot of interest in “physics-informed” Machine-Learning approaches

Where to inject the physics?

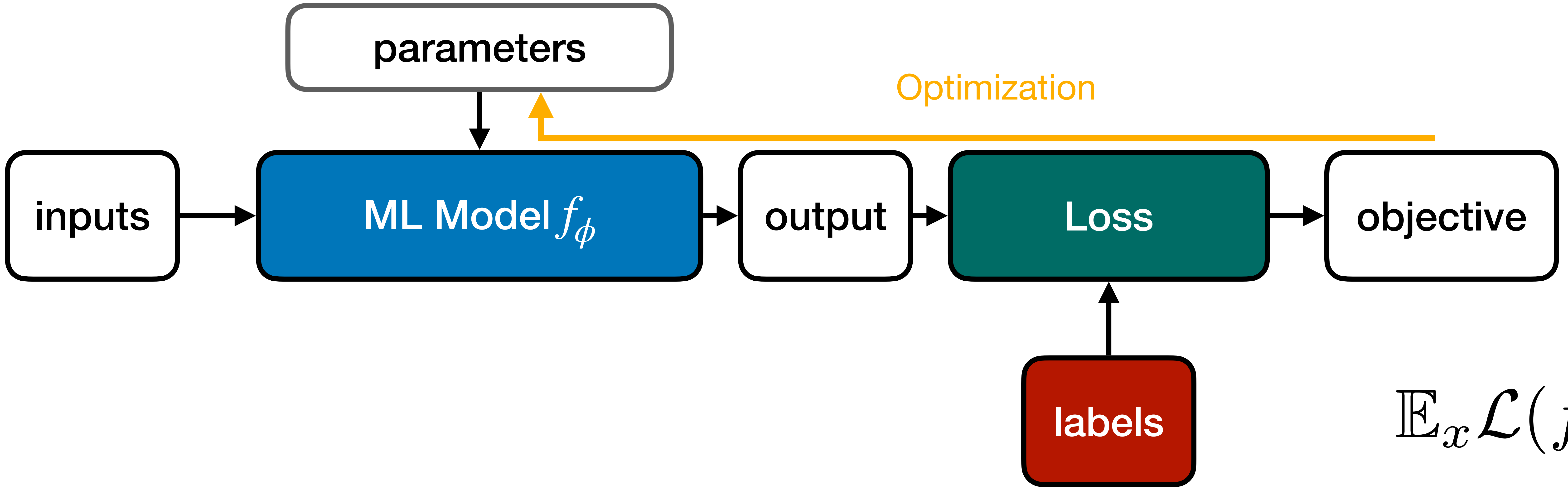


$$\mathbb{E}_x \mathcal{L}(f_\phi(x), y)$$



Ideally: inject physics domain knowledge in all areas of a ML system

How to inject the physics?



$$\mathbb{E}_x \mathcal{L}(f_\phi(x), y)$$

Ability to computation gradients of computer programs are a key mechanism to inject domain knowledge into ML

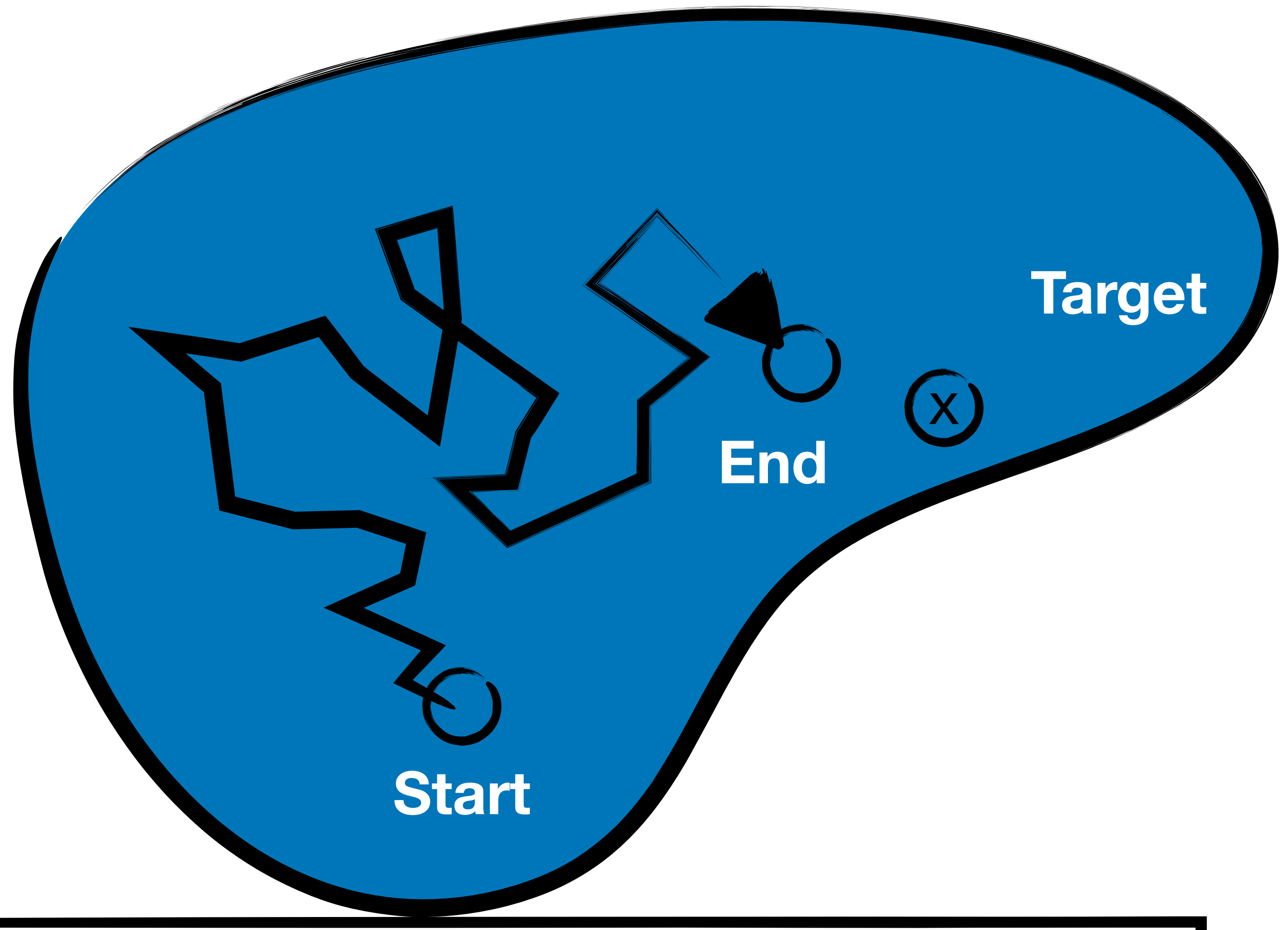
↳ **Differentiable Programming**

Gradient Based Optimization

Deep Learning is about searching through am extremely high-dimensional space

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\ \mathbb{E}_x \mathcal{L}(f_\phi(x), y)$$

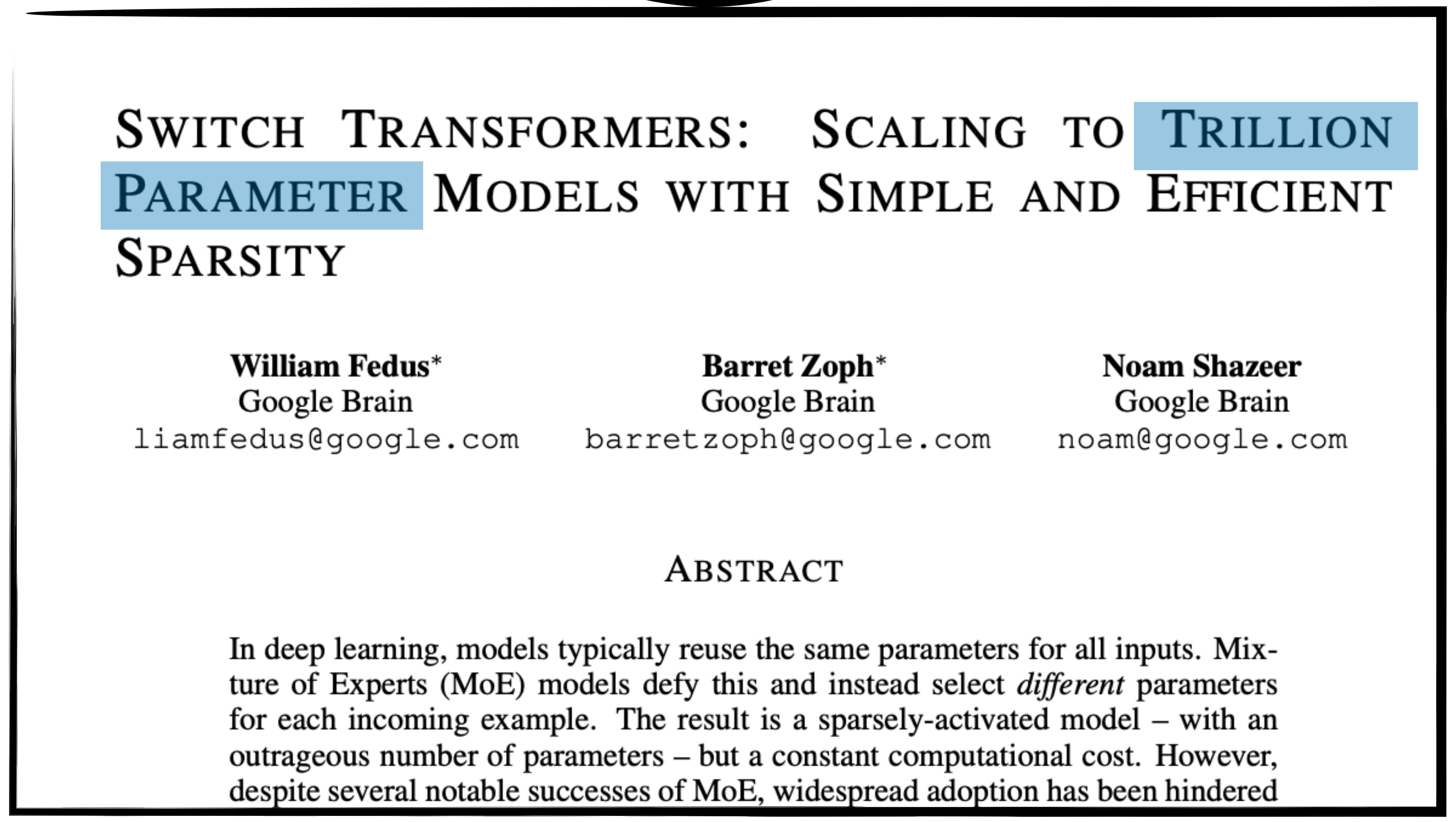
Space of Algorithms



Gradients with respect to the algorithm parameters are crucial in order to make this feasible at all.

Requires differentiable models & differentiable losses

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



SWITCH TRANSFORMERS: SCALING TO TRILLION PARAMETER MODELS WITH SIMPLE AND EFFICIENT SPARSITY

William Fedus* Google Brain liamfedus@google.com
Barret Zoph* Google Brain barretzoph@google.com
Noam Shazeer Google Brain noam@google.com

ABSTRACT

In deep learning, models typically reuse the same parameters for all inputs. Mixture of Experts (MoE) models defy this and instead select *different* parameters for each incoming example. The result is a sparsely-activated model – with an outrageous number of parameters – but a constant computational cost. However, despite several notable successes of MoE, widespread adoption has been hindered

arXiv:2101.03961

Automatic Differentiation

Automatic Differentiation: careful application of *chain rule to computer programs*

- exact gradients (as e.g. Mathematica), but low overhead
- available for many common programming languages

```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
```

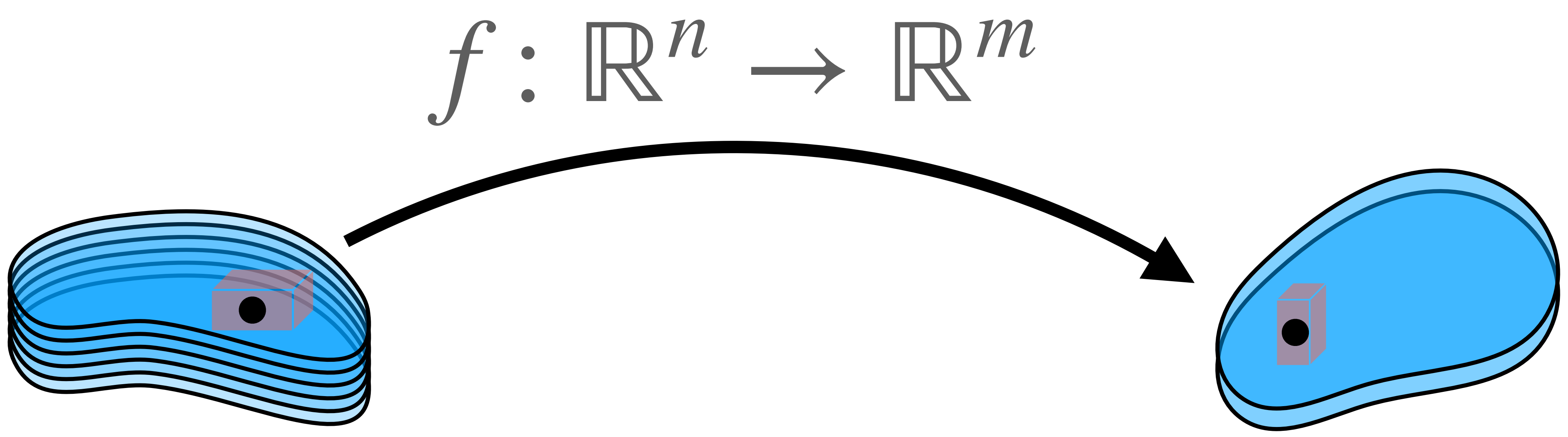
exact gradients!

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

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

TensorFlow JAX PYTORCH

... but also C++, Fortran, ...

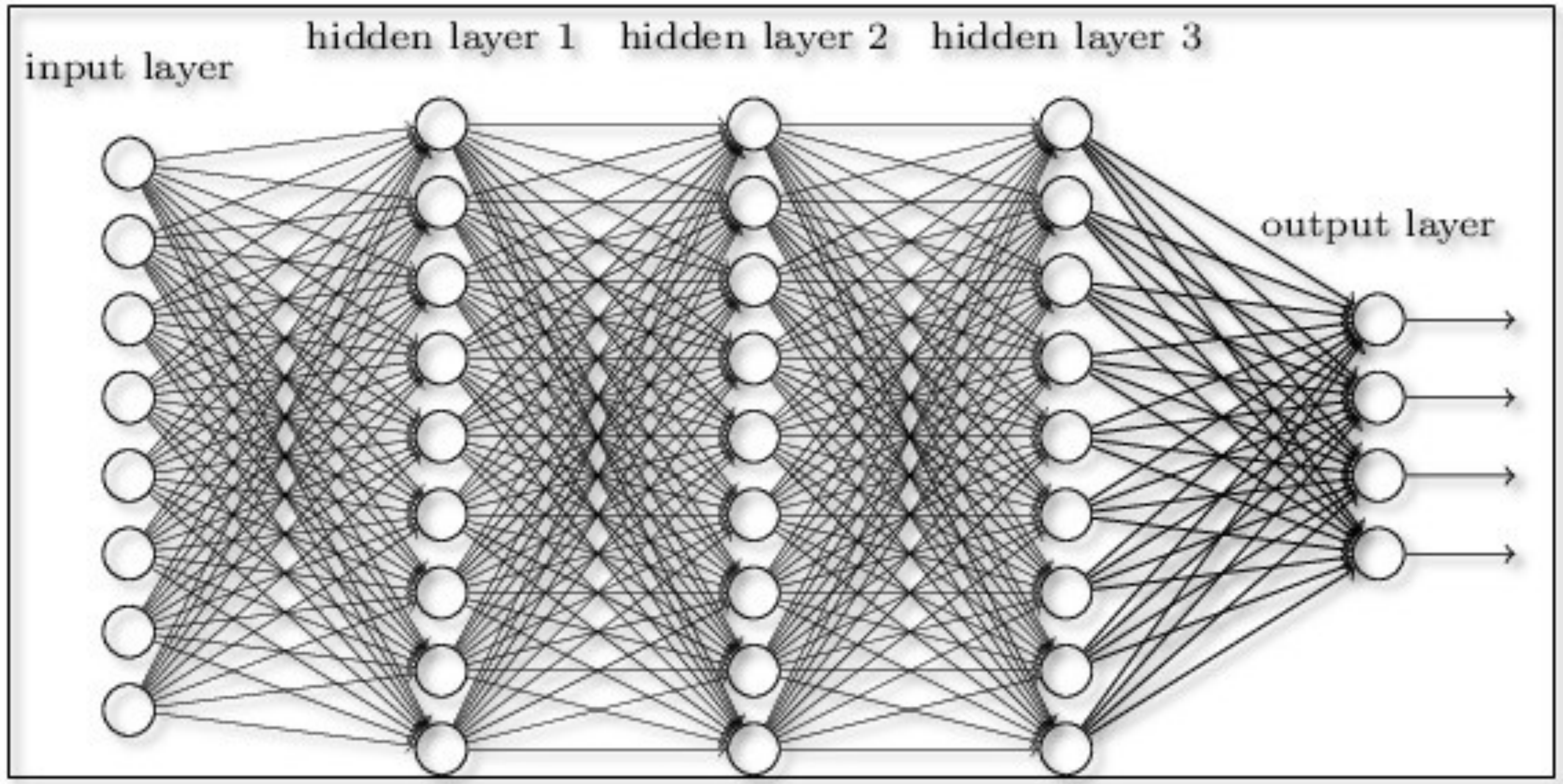


$$y = f(x) \qquad dy = J_f \, dx$$

Normal Program Output **Additional Program Output w/ AD**

What is the space of algorithms?

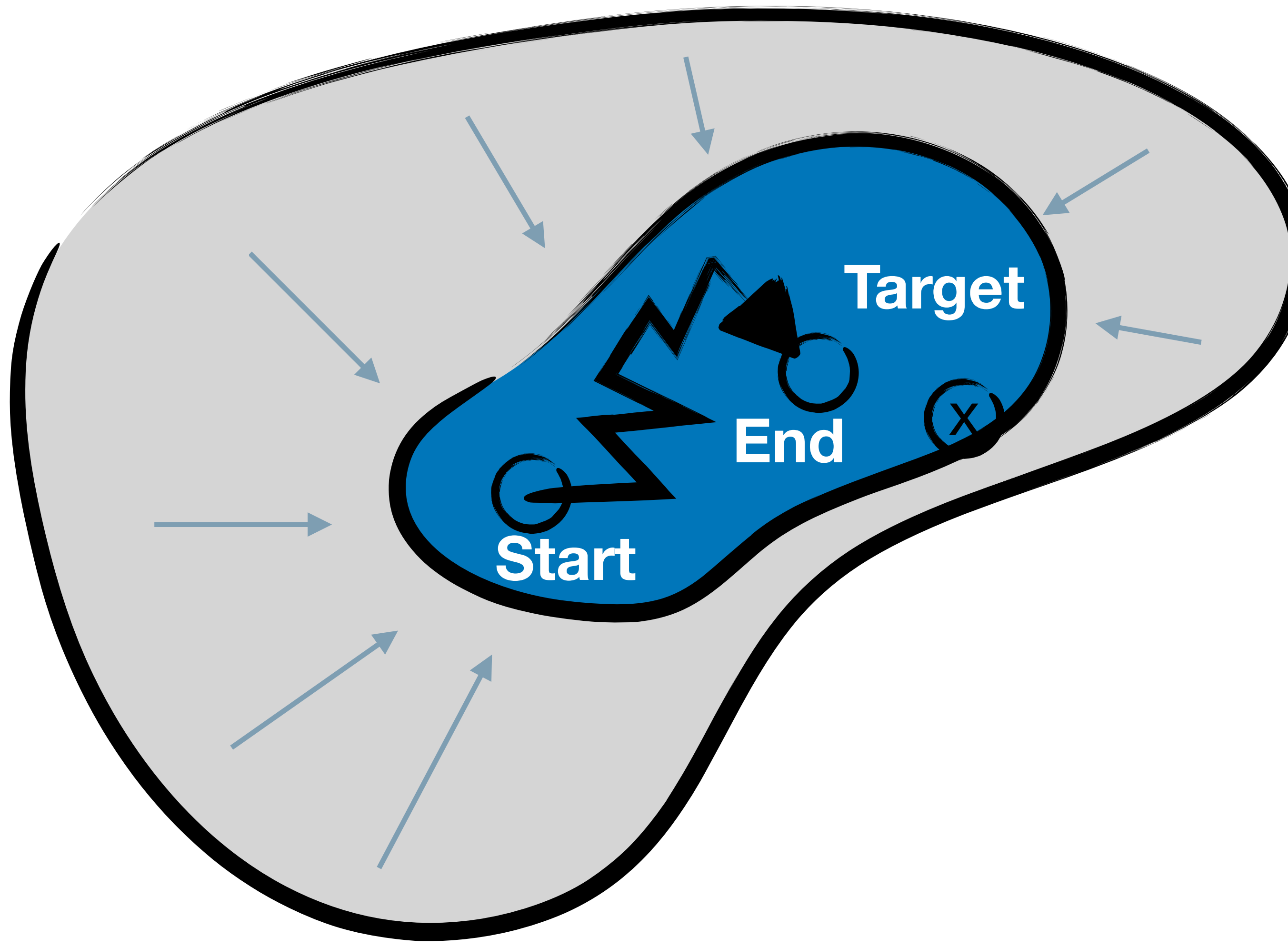
Classic Neural Network answer: a program without any structure



Generic layers that are easily differentiated

With sufficient data this can work.

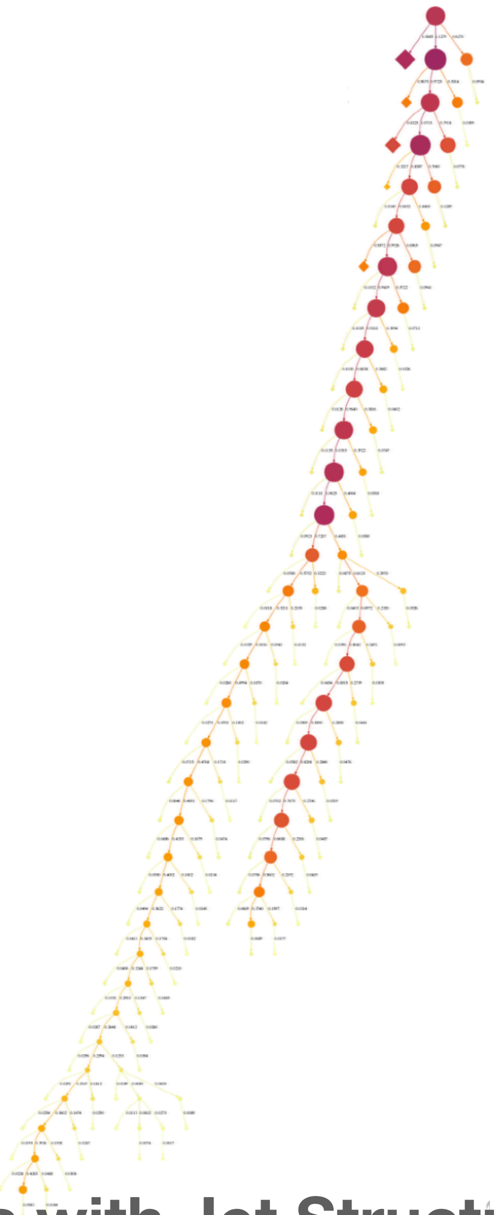
Inductive Bias



Architectures: By imposing structure on the program we can **bias learning towards sensible solutions**

- more interpretable & data-efficient

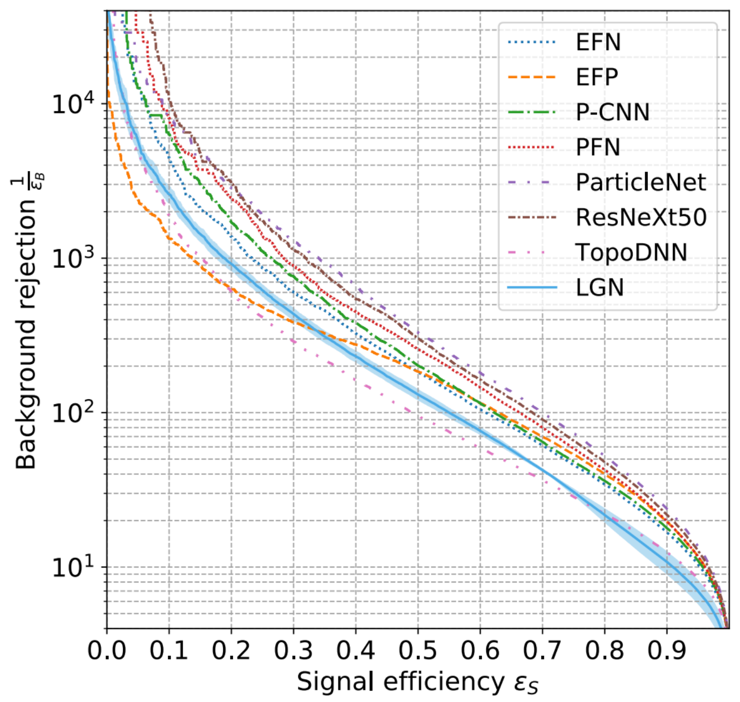
Constraint: program must stay differentiable wrt. parameters to allow gradient-based optimization



Nets with Jet Structure

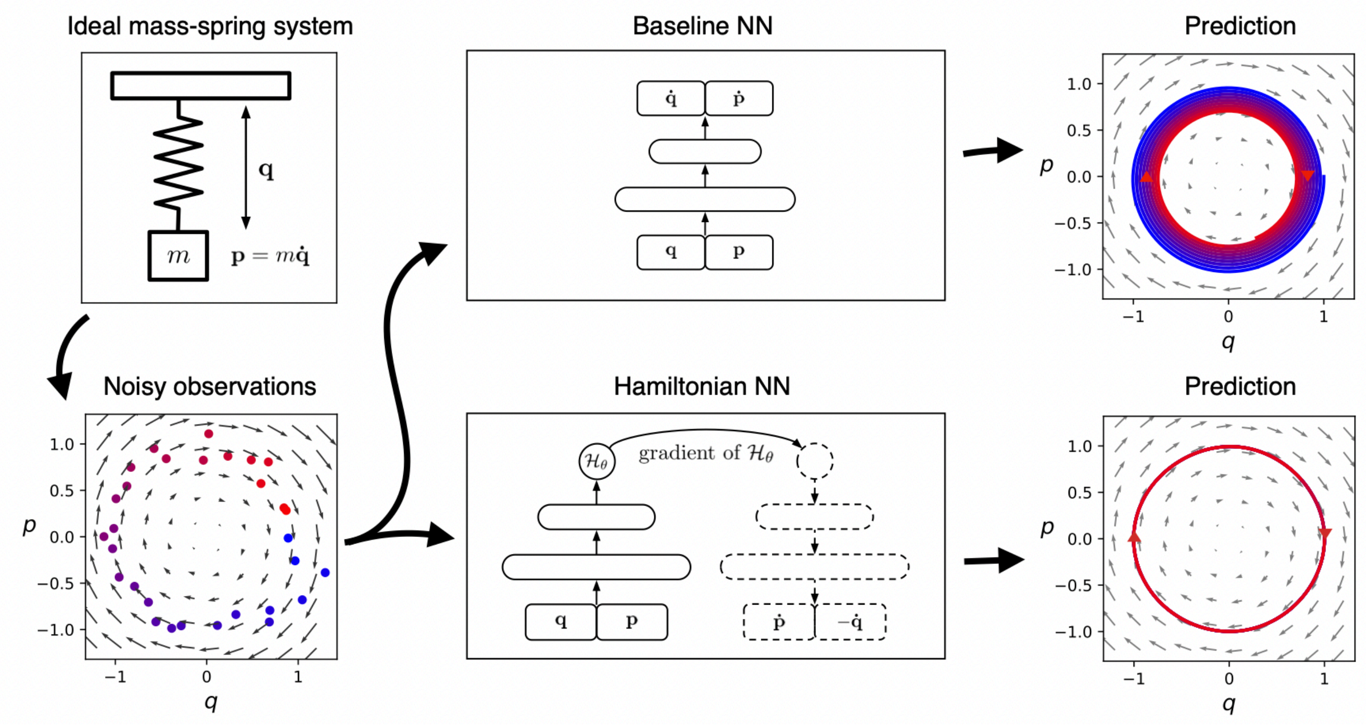
arXiv:1702.00748

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 ± .001	0.964 ± 0.018	435 ± 95	4.5k



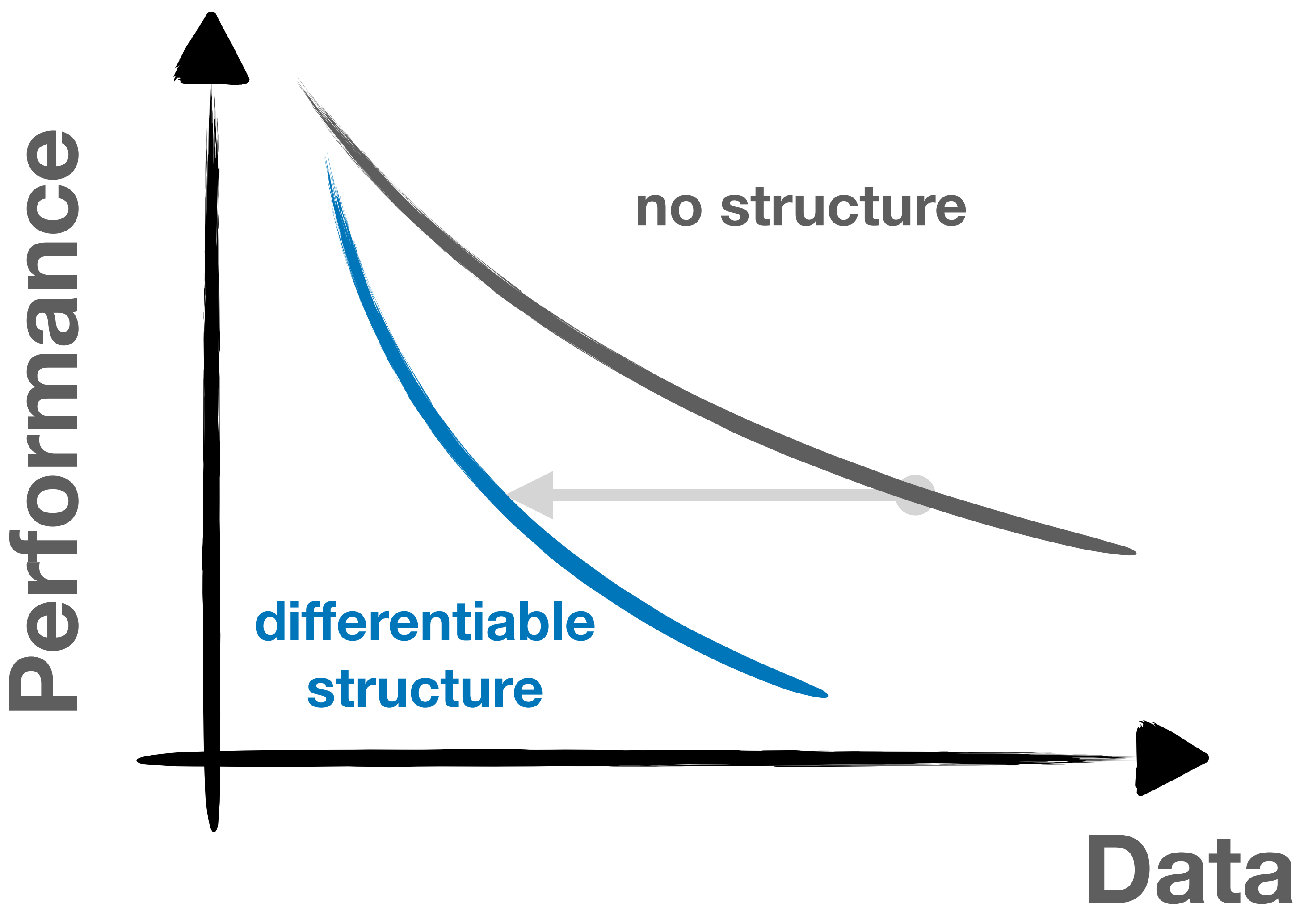
Lorentz-Invariance

arXiv:2006.04780



Hamiltonian Neural Nets

arXiv:1906.01563



Why stop there?

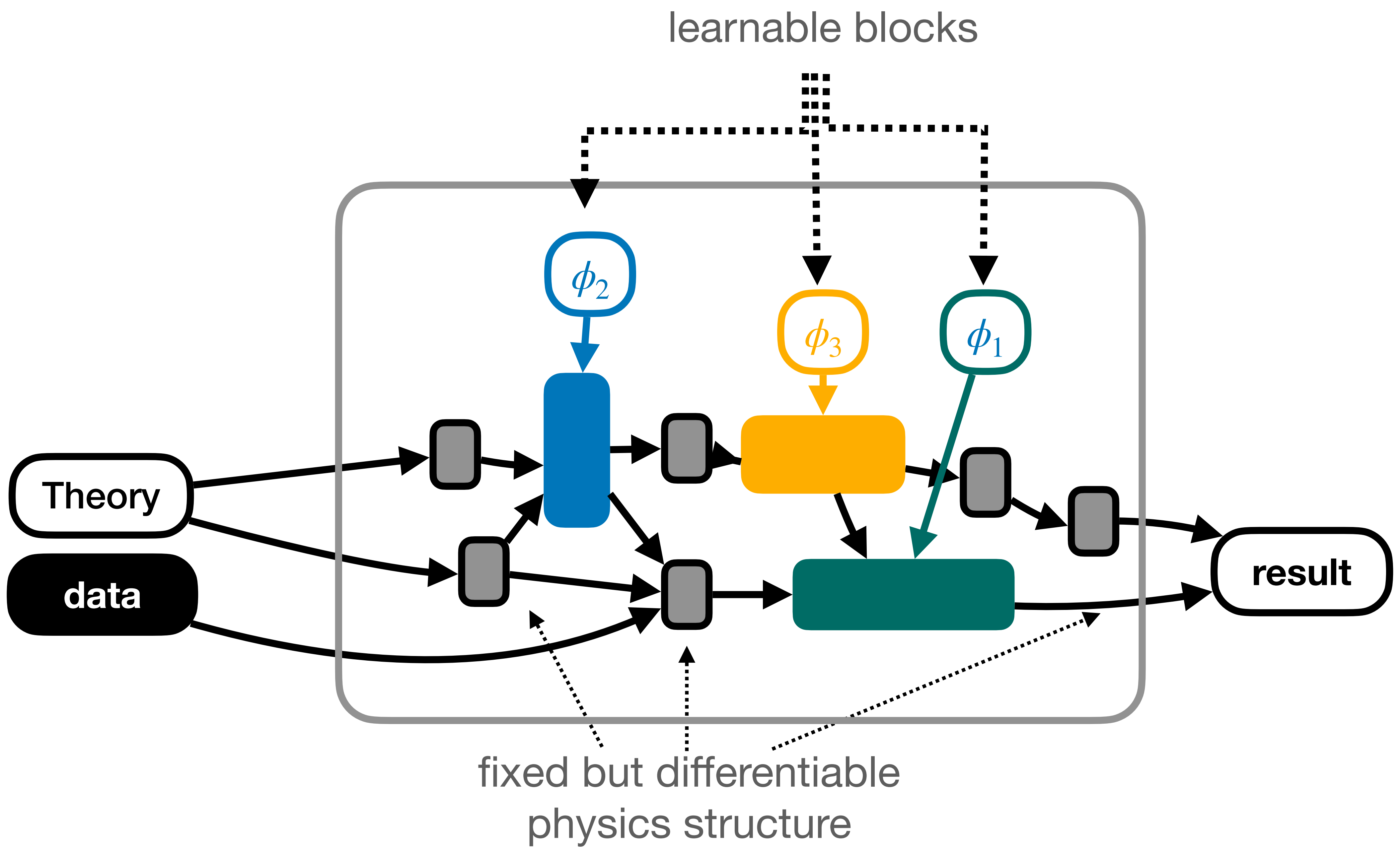
We already have a lot of code and structure that encodes our physics intuition

- Simulation, Tracking, Calorimetry, Particle Identification, Event Observables, ..

Instead of adding structure to a neural networks (symmetries, …) we can try to make our existing already-structured programs / logic differentiable

The differentiable programming POV

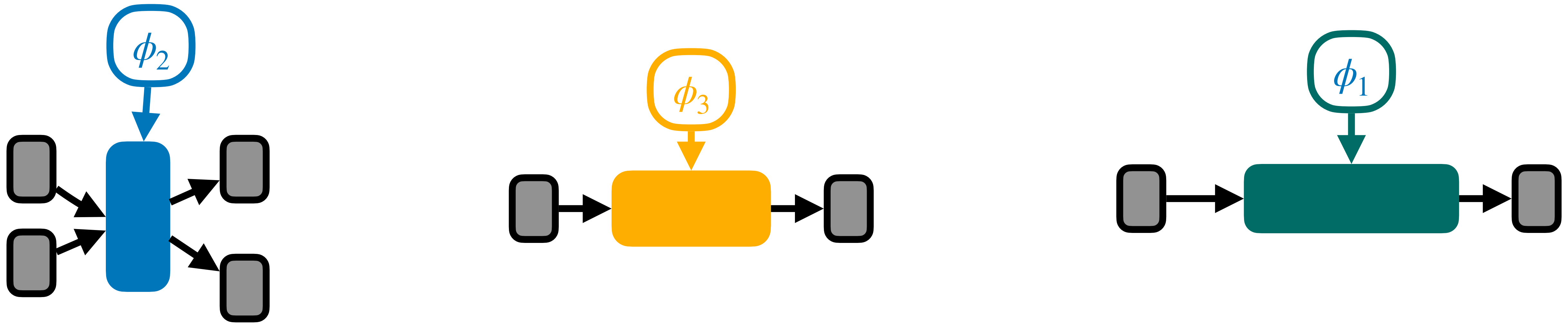
- enforce structure where we want it
- let ML fill in the blanks
- joint end-to-end optimization



What do we get from this?

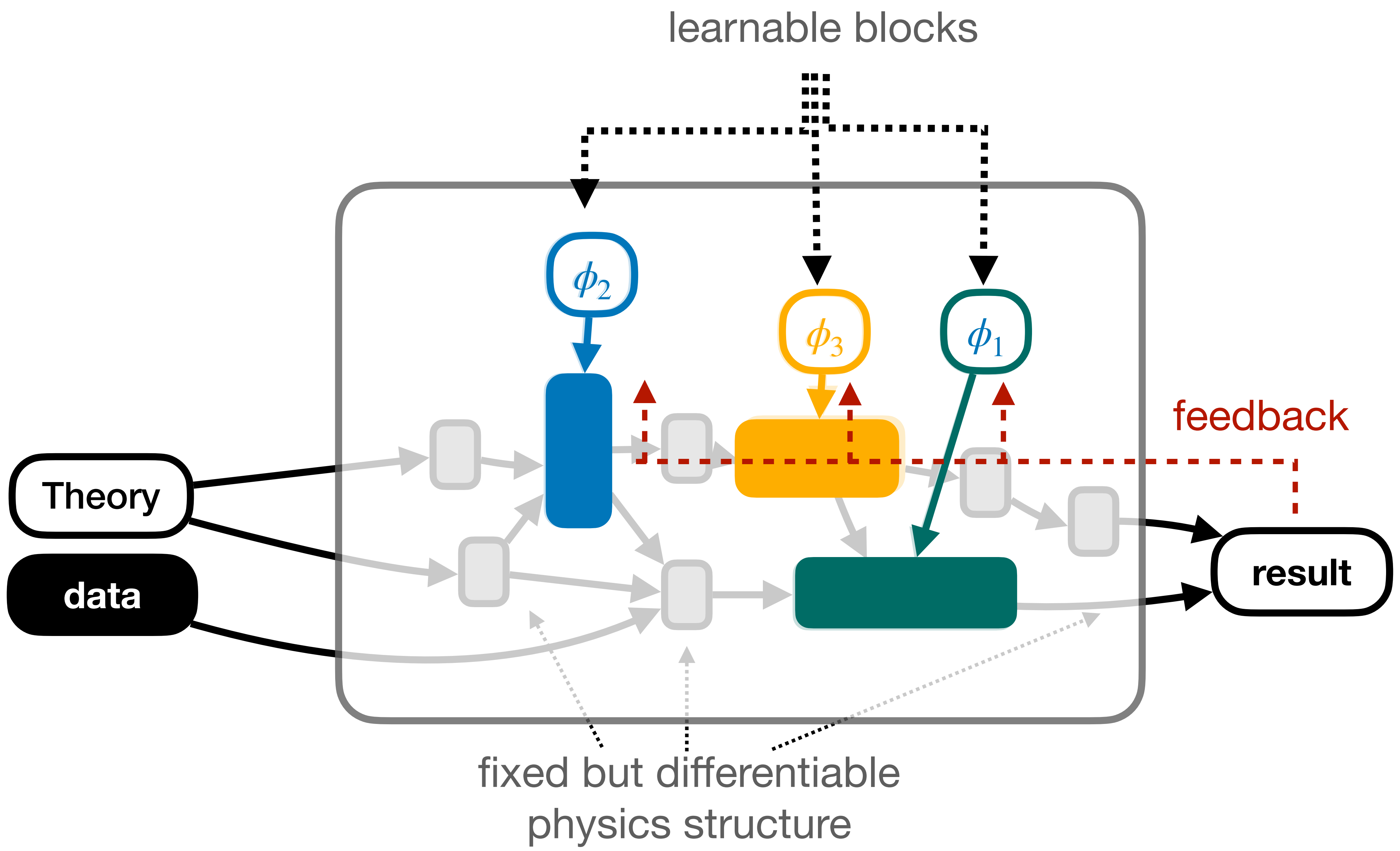
Currently we enforce physics by **compartmentalizing the ML components**

- train tracking, <u>then</u> particle ID, <u>then</u> analysis discriminants



With end-to-end differentiable programming we can guide **low-level algorithms** with **high-level feedback**

e.g. optimize reconstruction on final physics performance

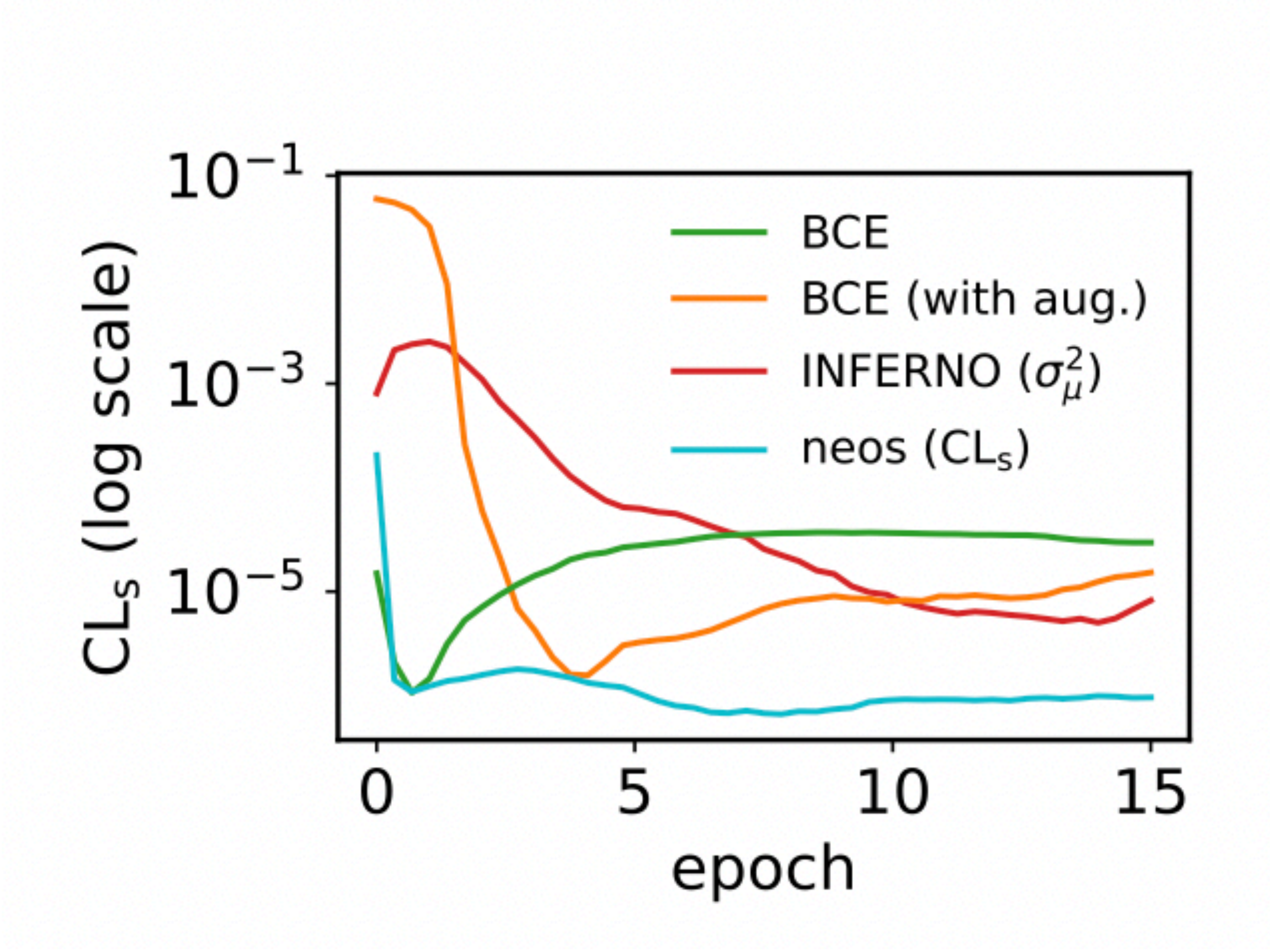


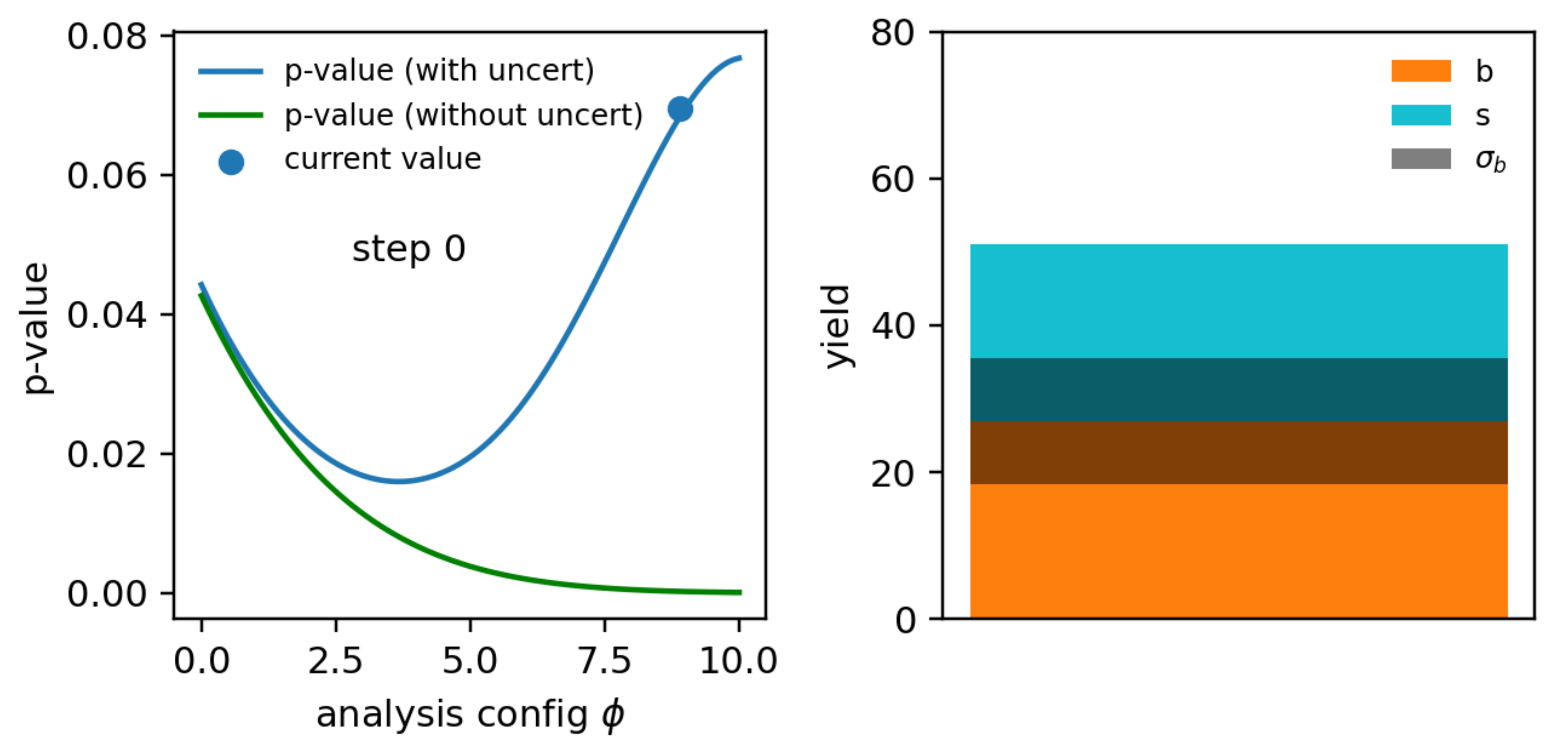
Example: Systematics-Aware Neural Networks

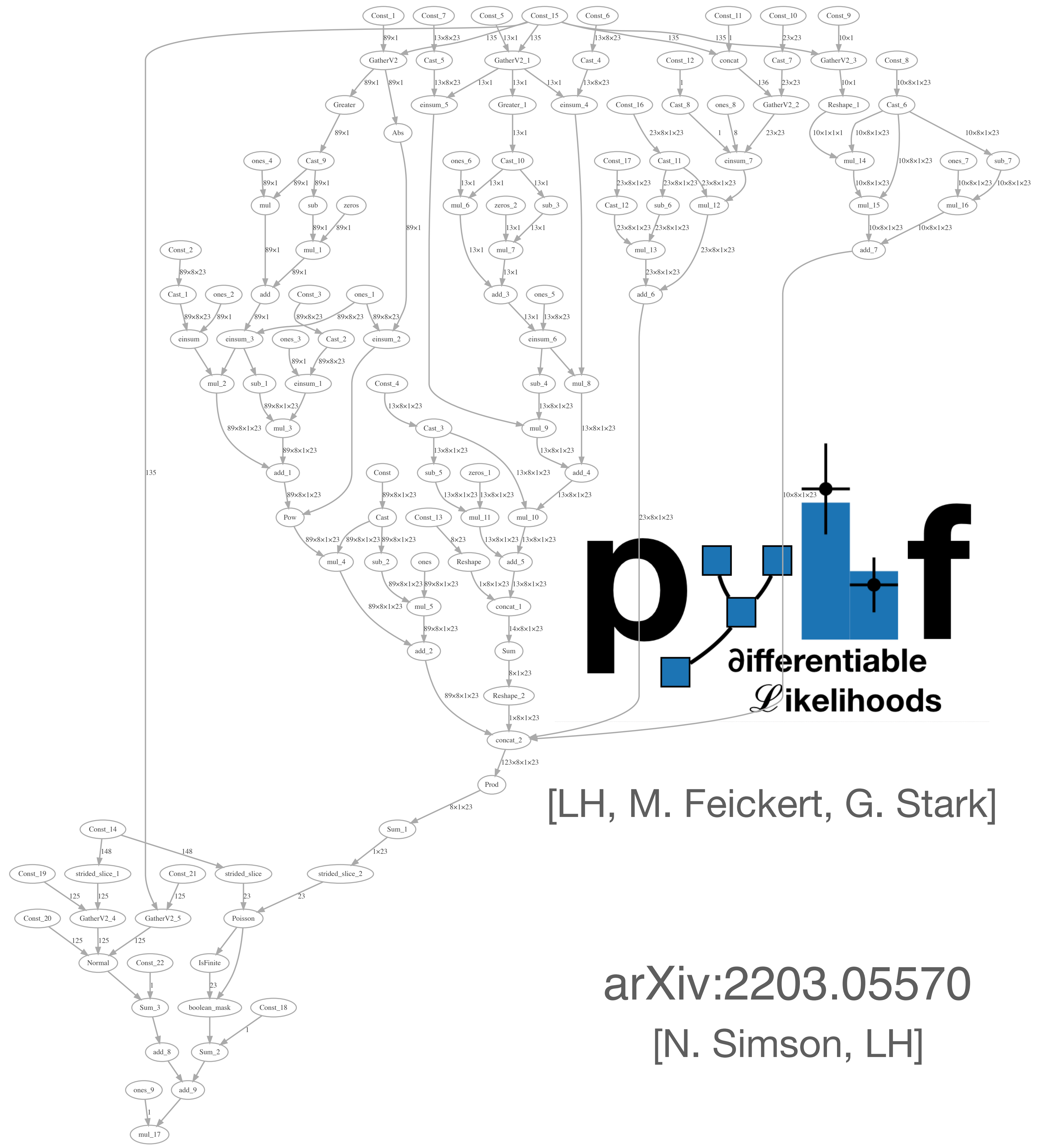
differentiable
but not a neural net!

Instead of optimizing on a proxy non-physics goal we can optimize on e.g. actual physics sensitivity

“smarter loss” thanks to a fully differentiable statistical analysis incl. systematics modelling, profiling.







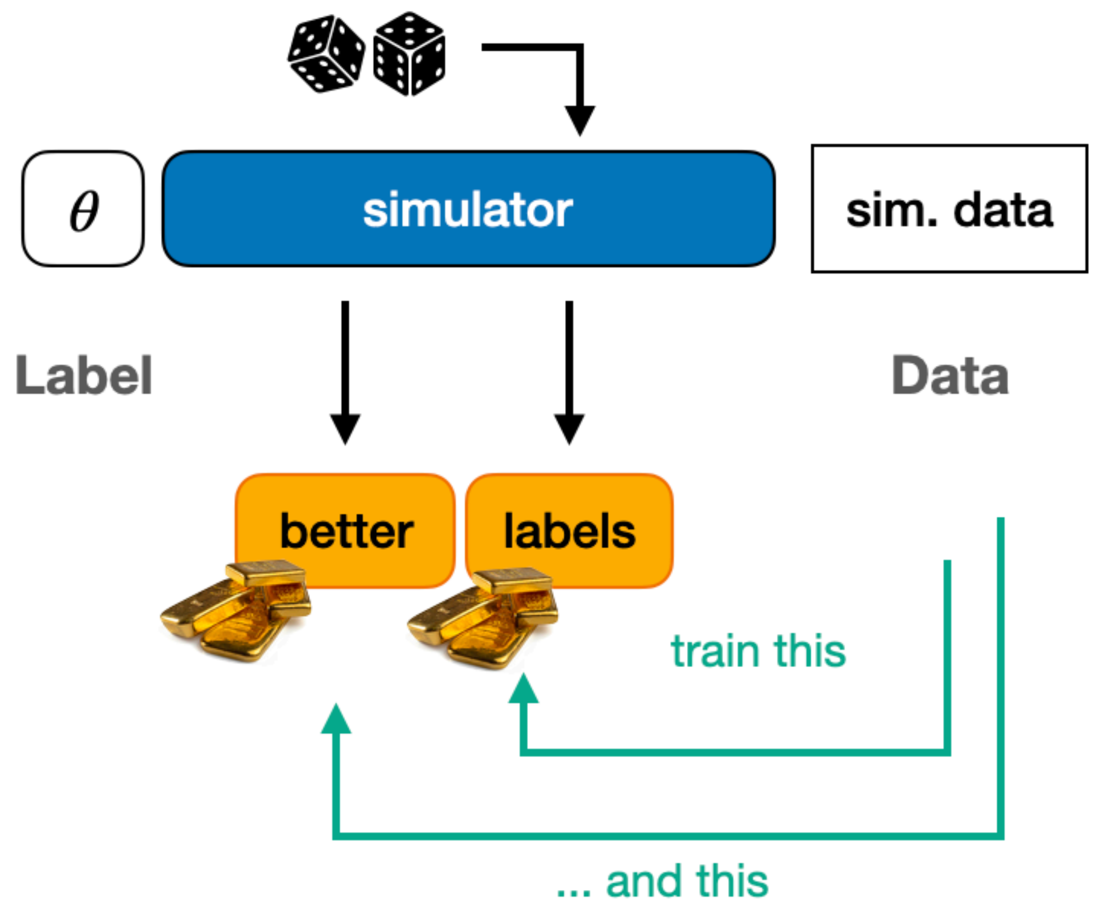
[LH, M. Feickert, G. Stark]

arXiv:2203.05570
[N. Simson, LH]

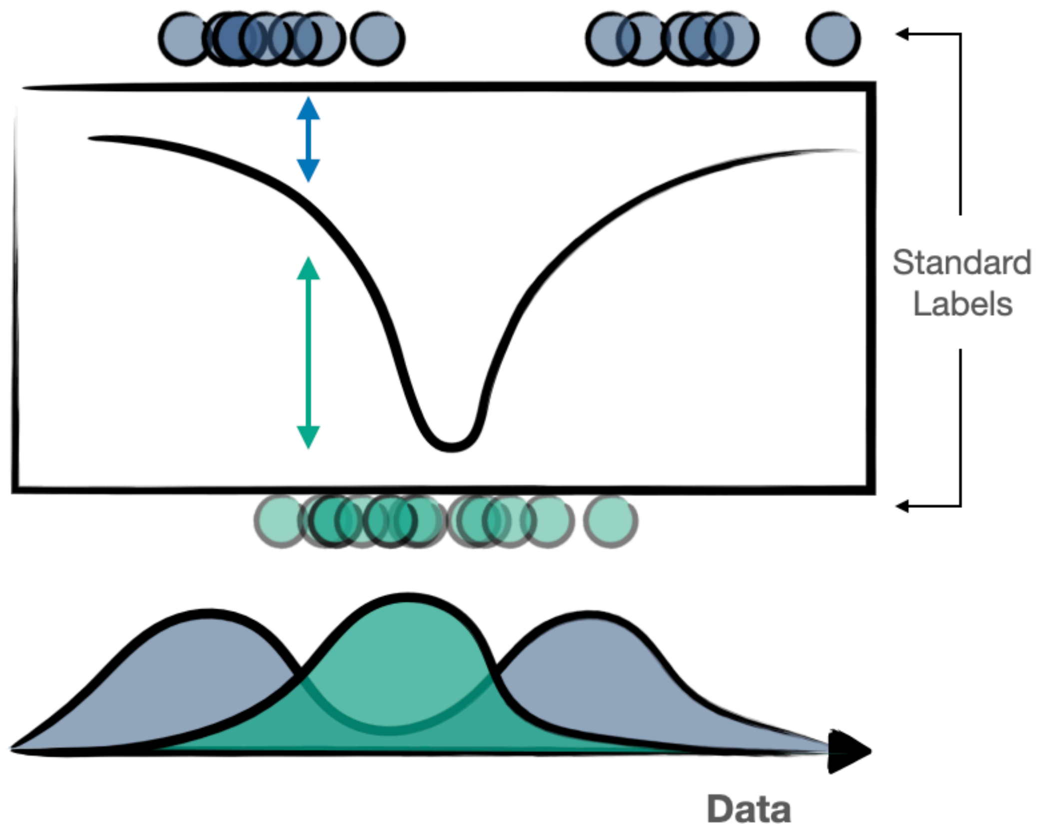
arXiv:1806.04743
P. de Castro, T. Dorigo]

Example: Gradient-Based Labels

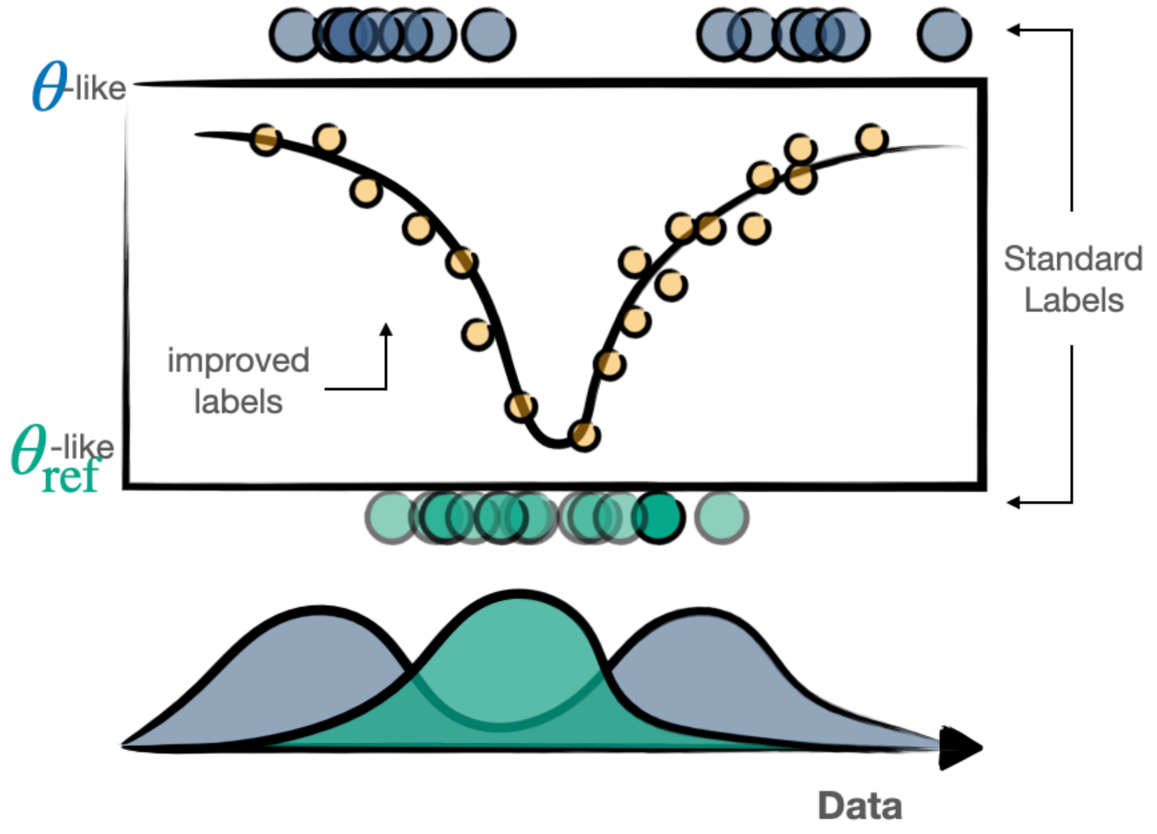
Gradients are also key in order to improve labels. Intuitive: The more information you have about a target function the better



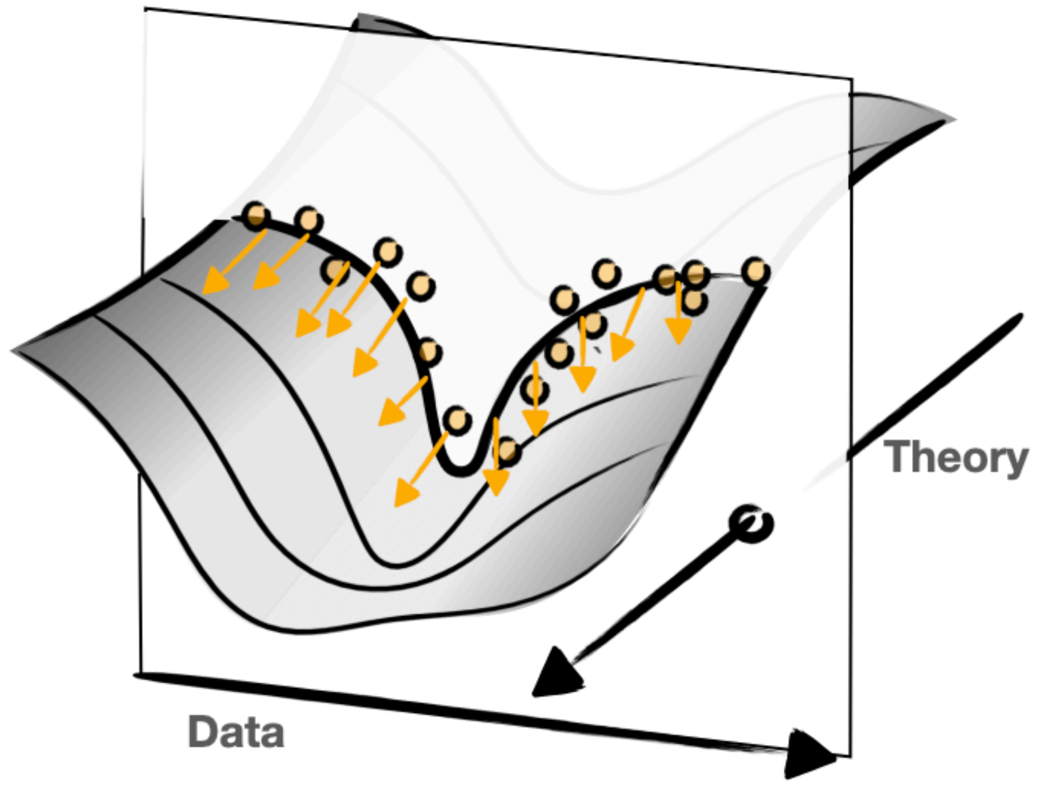
Mining Gold:
Extract labels from a (differentiable) simulator for density ratio estimation



θ



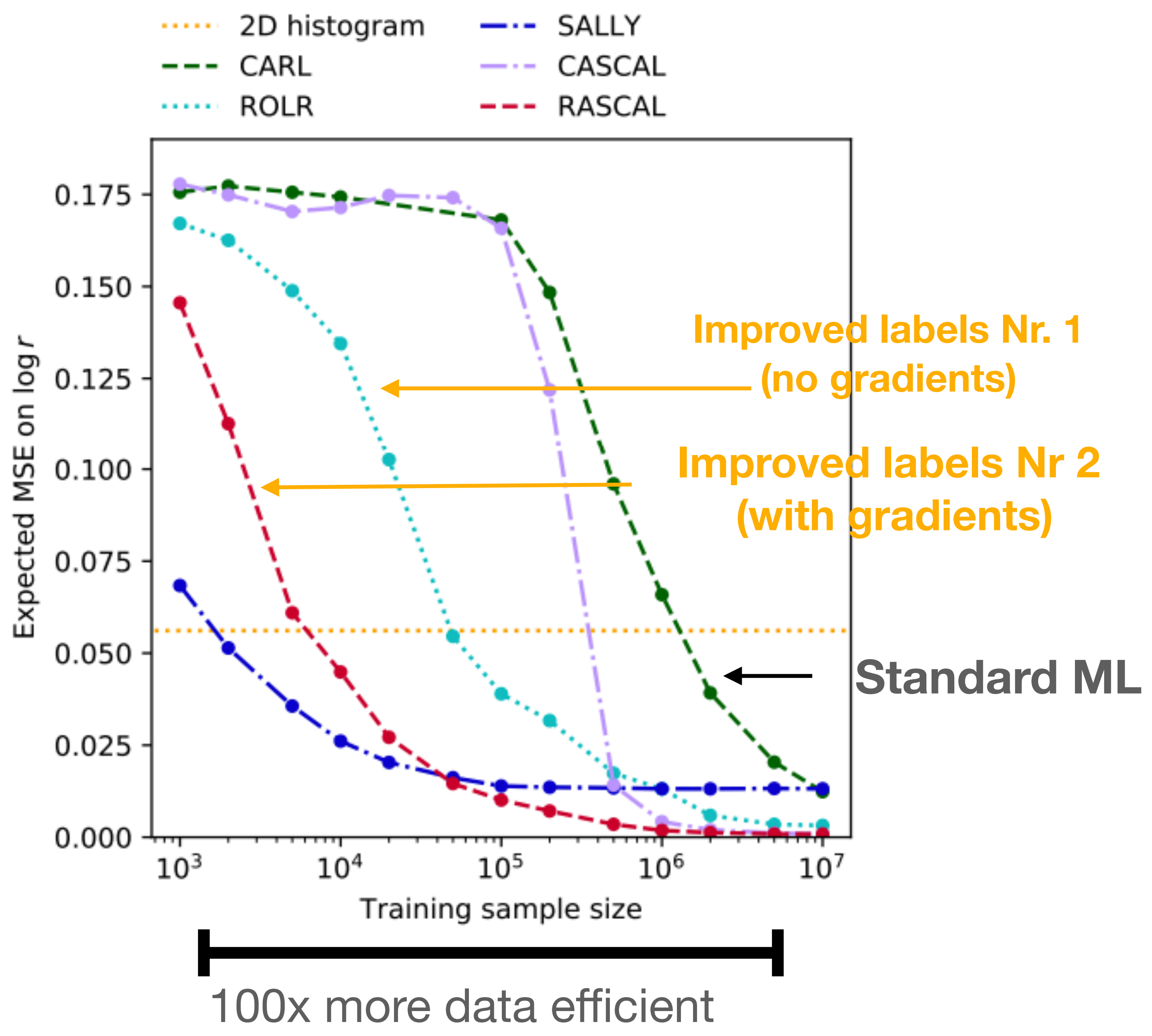
$\log p(x, z \,|\, \theta)$

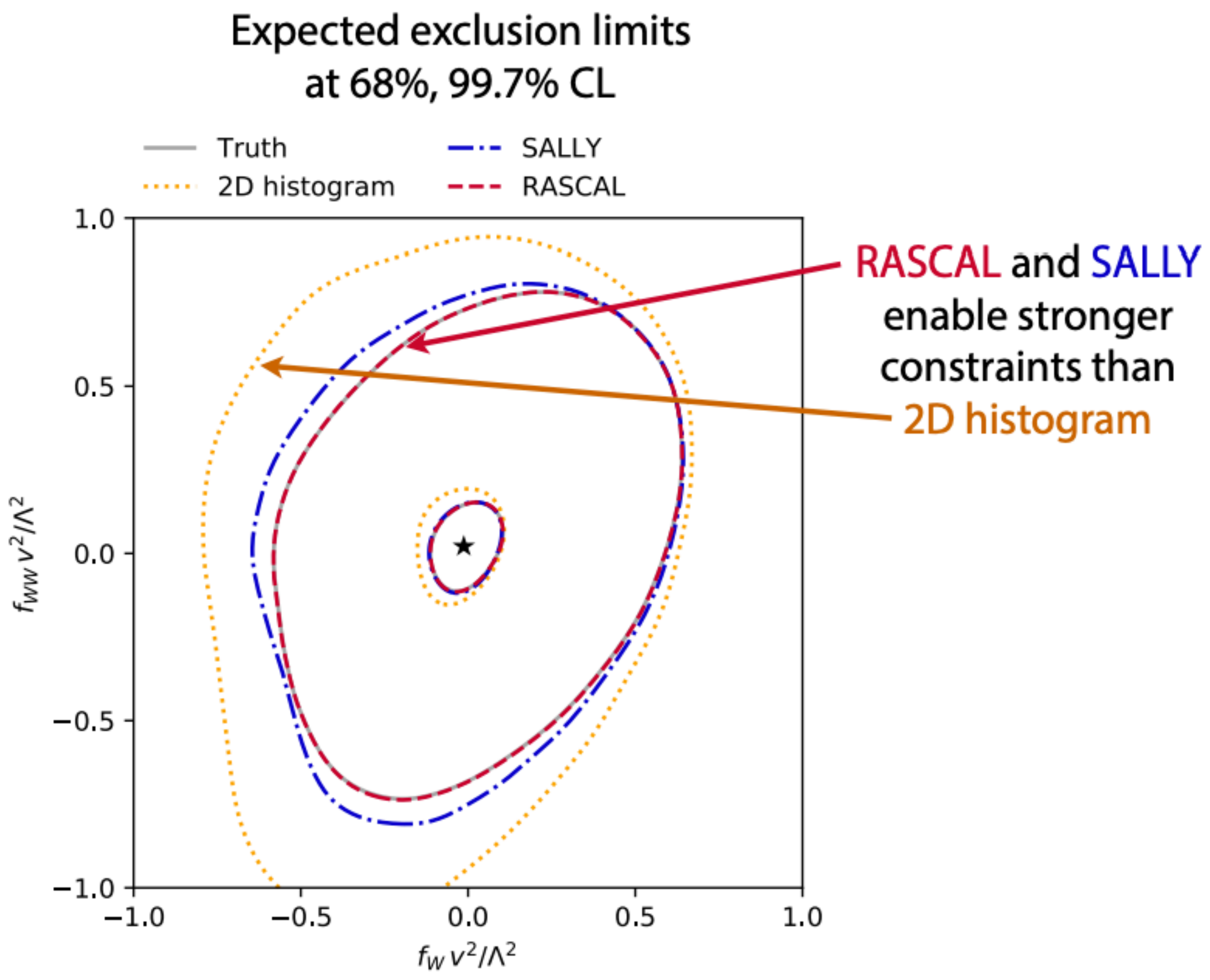


$\nabla \log p(x, z \,|\, \theta)$

Example: Gradient-Based Labels

Improved gradient labels can improve physics a lot!





But requires differentiable Matrix Elements $\nabla_\theta |\mathcal{M}(x,\theta)|^2$

Differentiable MadGraph: madjax

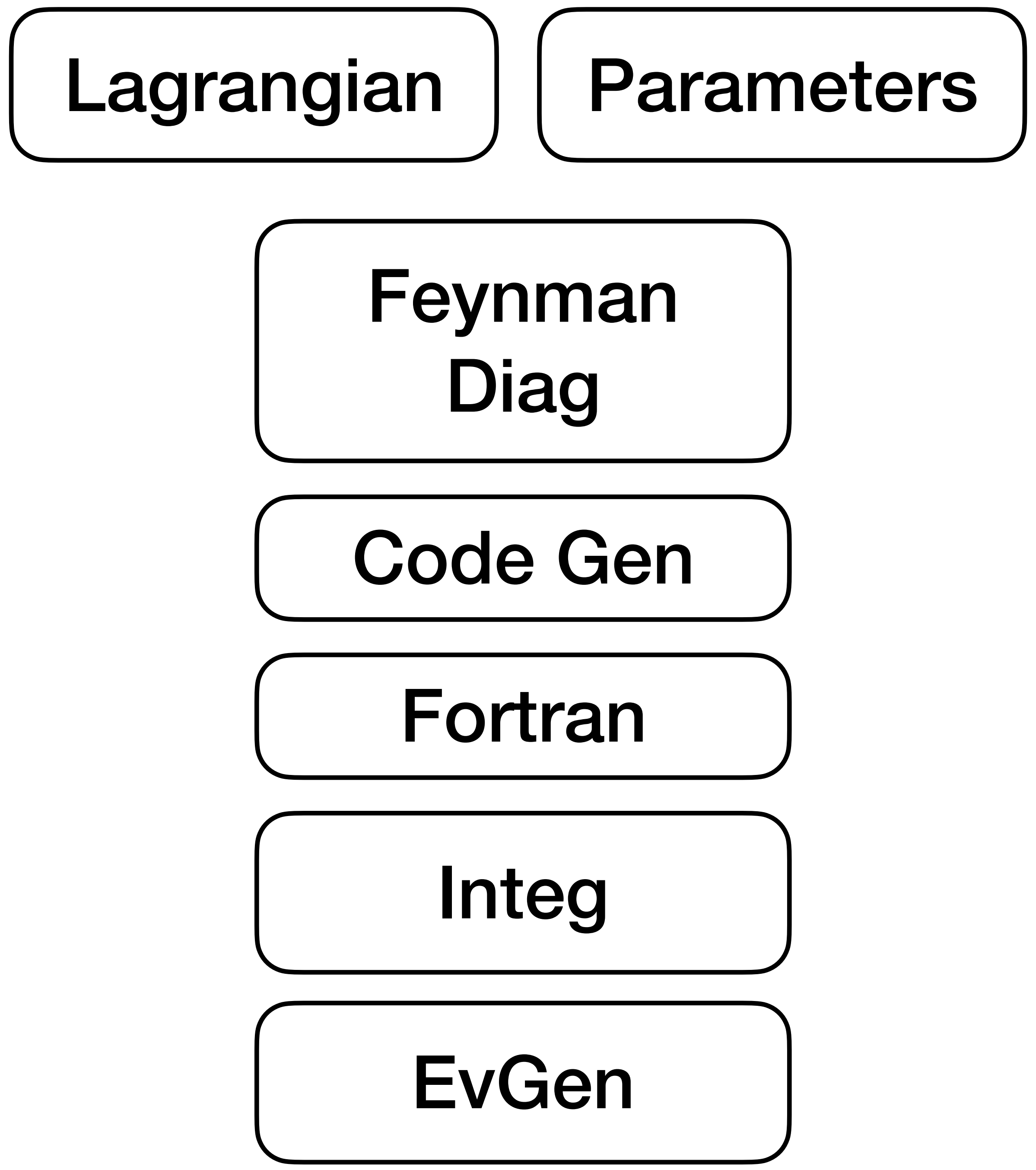
- General purpose Matrix Element Generator
- Default Choice for BSM Searches at LHC

Standard Simulator workflow:
Given a model, generate code to evaluate MEs

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

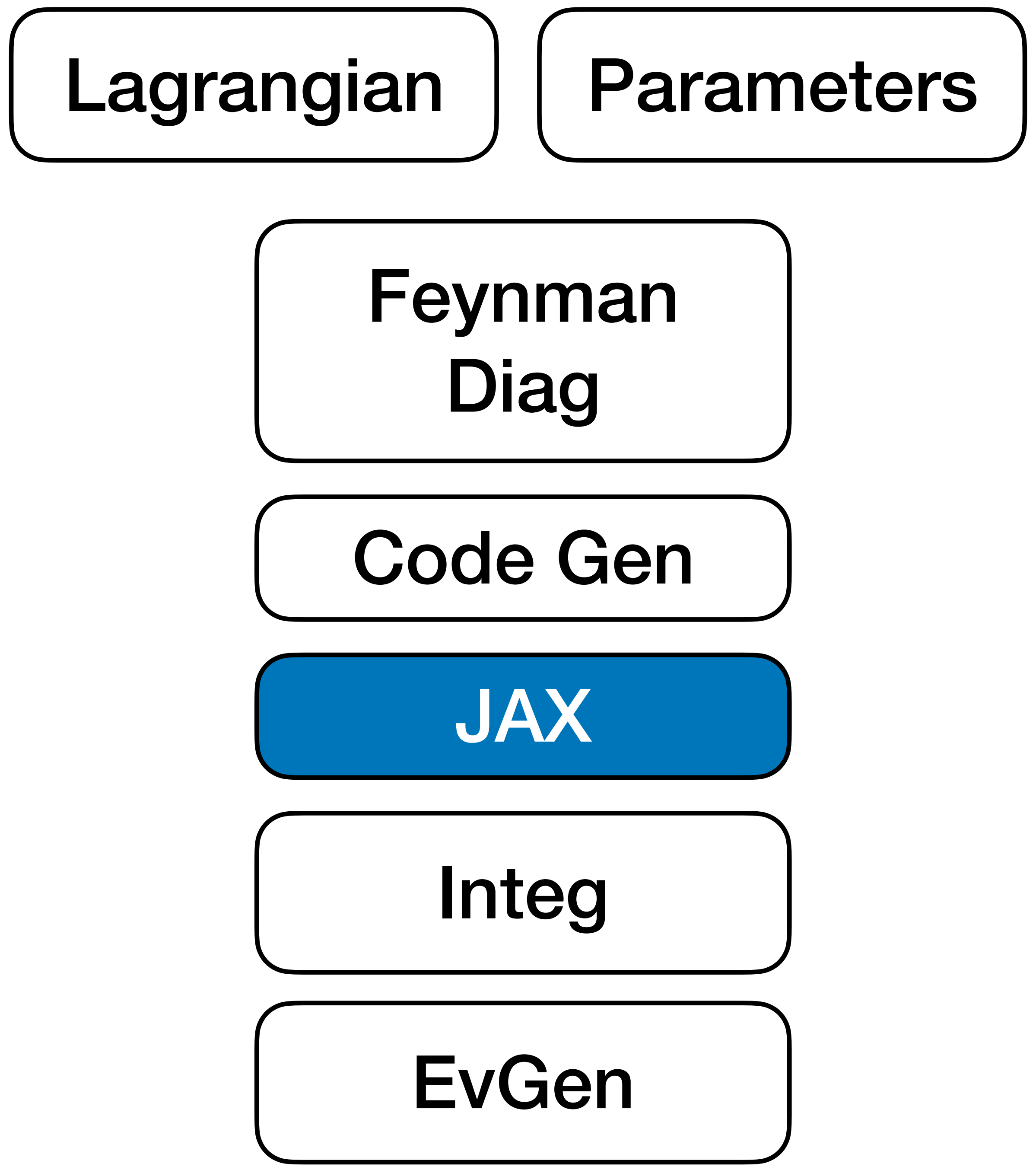
σ un-normalized pdf $\rightarrow$ $p(x|\theta) = \frac{1}{Z(\theta)} p(x|\theta)$

↑ from MC integration



[LH, Kagan] (WIP)

Differentiable MadGraph: madjax



- General purpose Matrix Element Generator
- Default Choice for BSM Searches at LHC

Idea:
Given a model, generate differentiable code to evaluate MEs

Automatically delivers additional physics information useful for downstream tasks

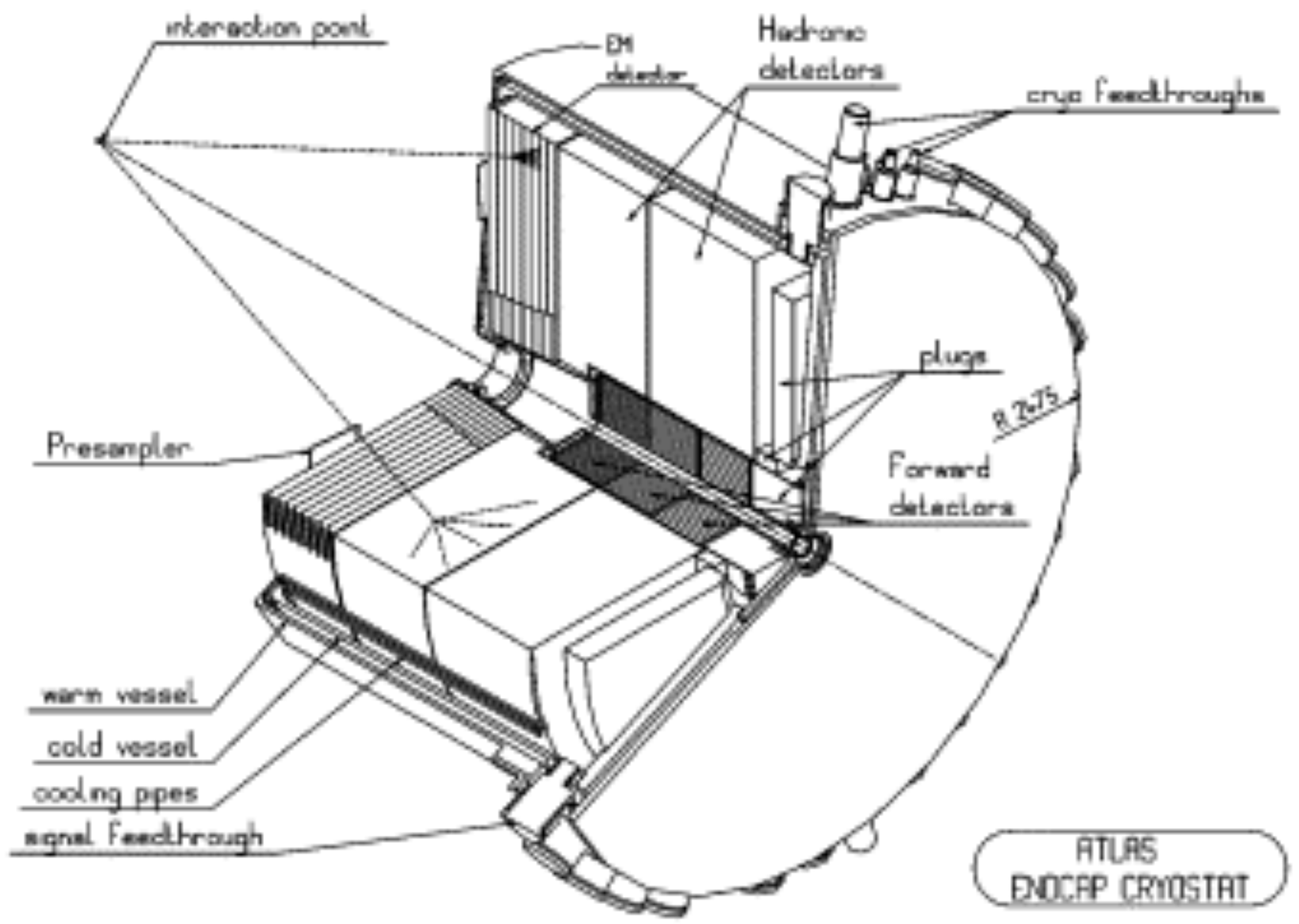
$\sigma(x, \theta)$	$\nabla_x \sigma(x, \theta)$	$\nabla_\theta \sigma(x, \theta)$
Matrix Elements	**Phase-space derivatives**	**Theory Landscape derivatives**

[LH, Kagan] (WIP)

Example: Differentiable Design Optimization

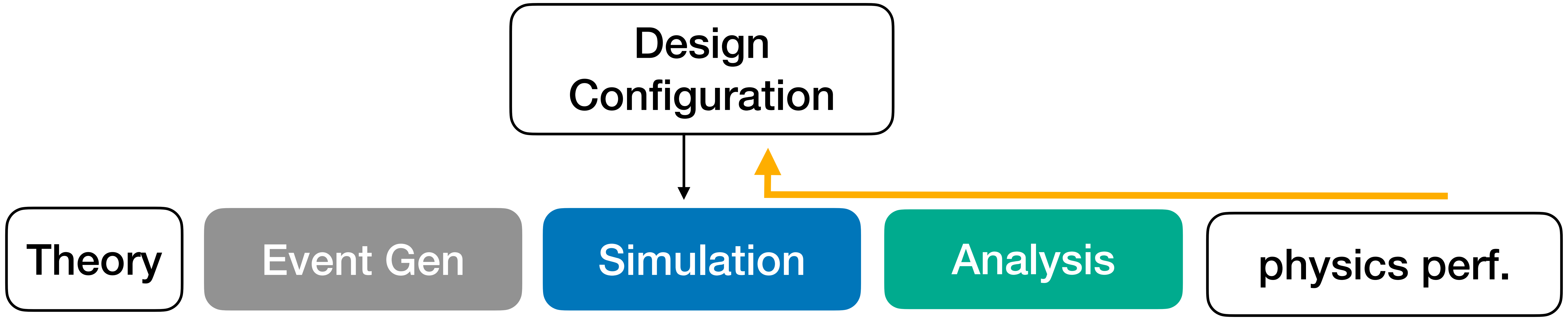
One of the most important optimization problems in physics is designing the detector itself

- very high-dimensional (many modules, …)
- many trade-offs not obvious at detector level but dictated by downstream physics goals

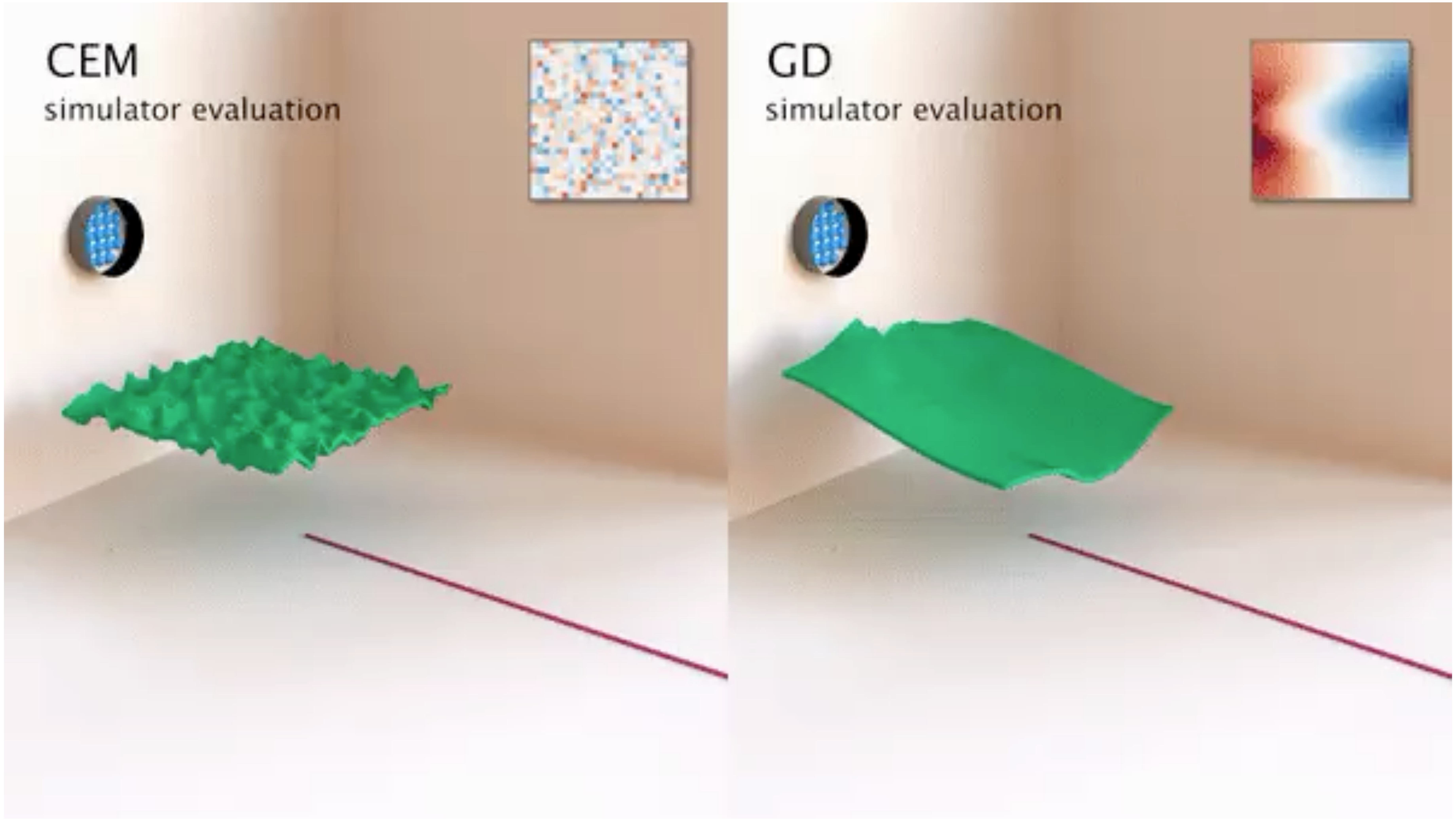


Idea: Can you use gradient-based optimization?

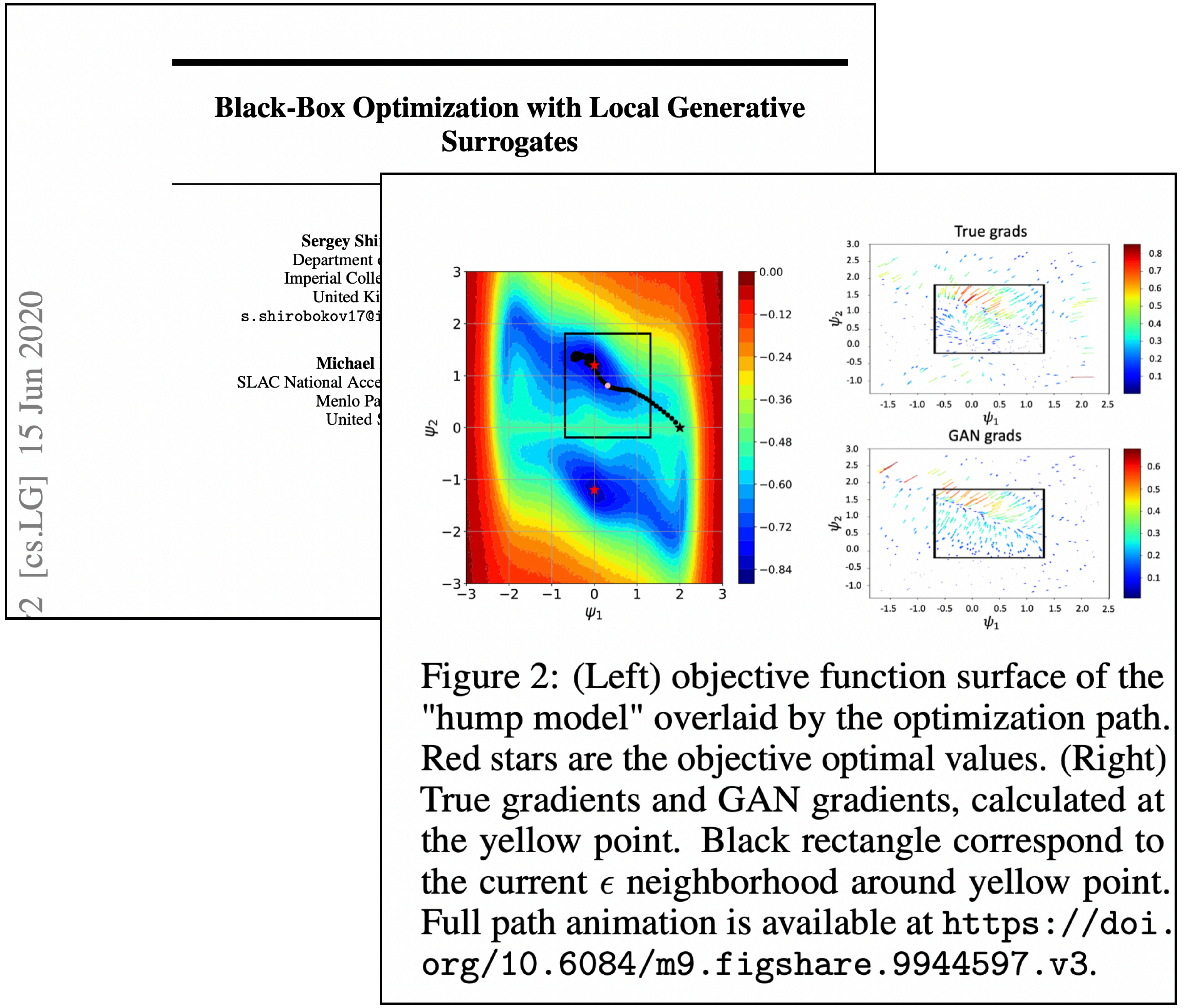
- **very ambitious idea, but potentially big payoff**



Successful Examples from outside of HEP



Successful Examples from inside of HEP

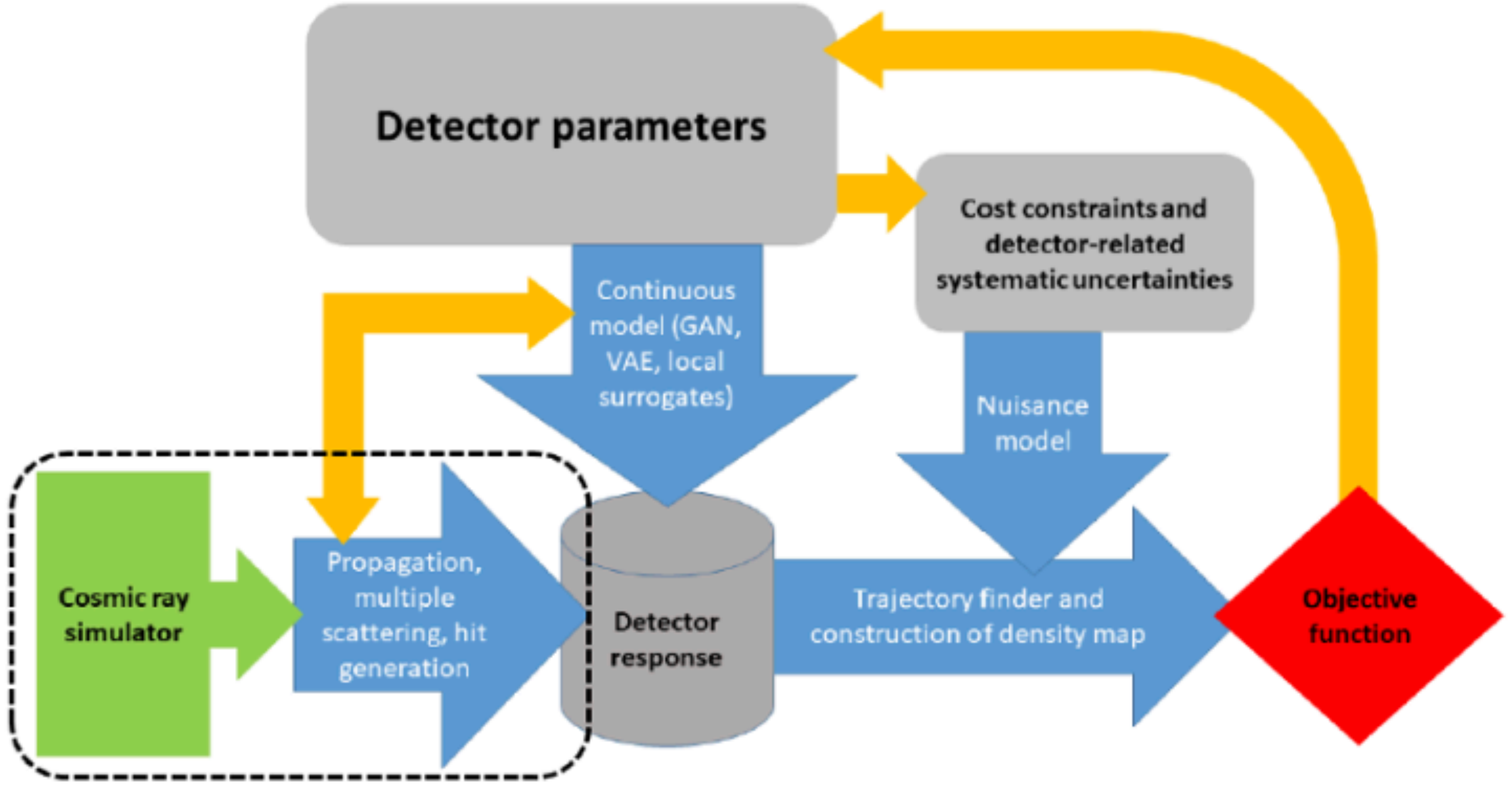

2 [cs.LG] 15 Jun 2020

Black-Box Optimization with Local Generative Surrogates

Sergey Shi
Department
Imperial Colle
United Ki
s.shirobokov17@i

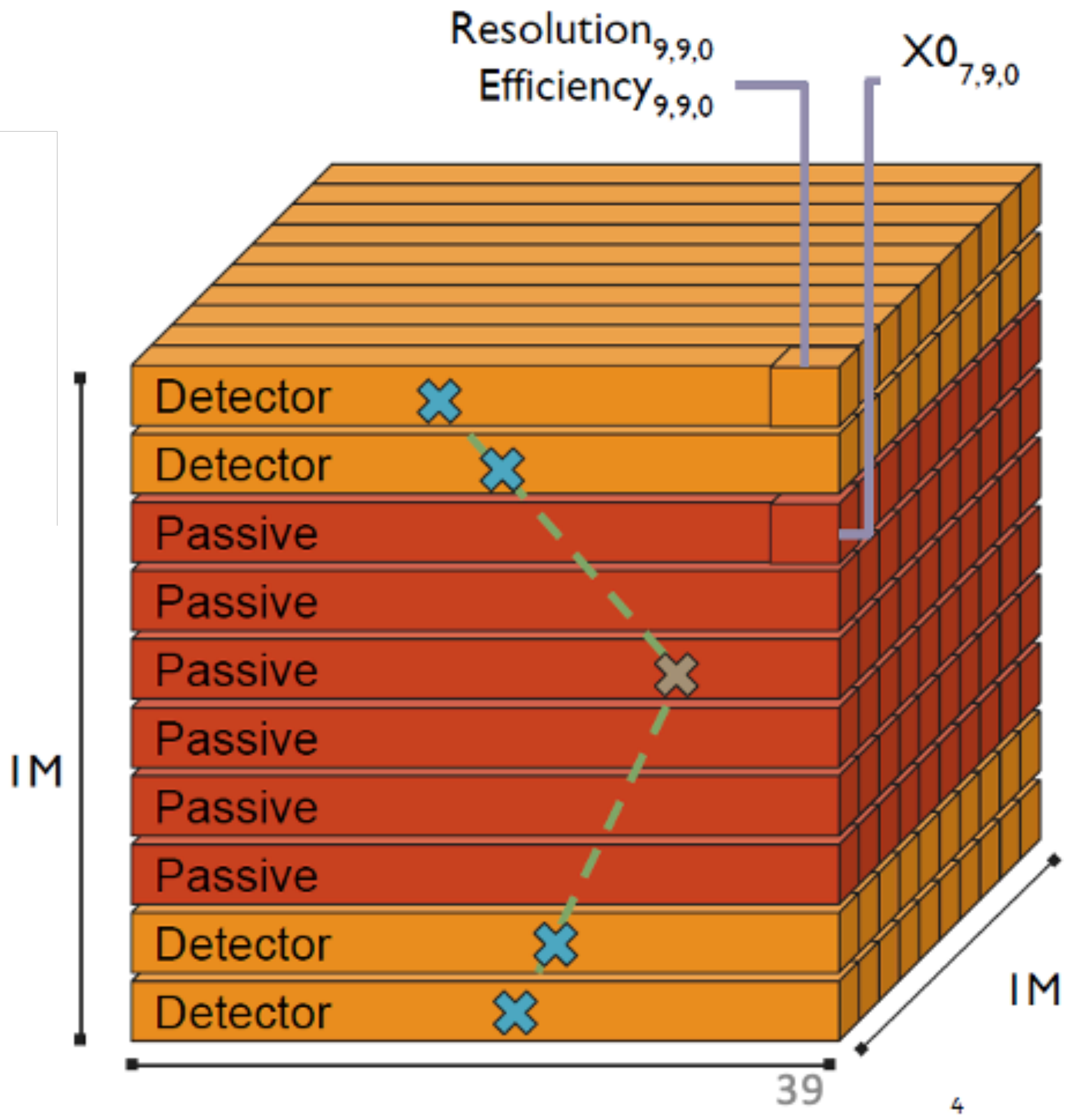
Michael
SLAC National Acce
Menlo Pa
United

Figure 2: (Left) objective function surface of the "hump model" overlaid by the optimization path. Red stars are the objective optimal values. (Right) True gradients and GAN gradients, calculated at the yellow point. Black rectangle correspond to the current ϵ neighborhood around yellow point. Full path animation is available at `https://doi.org/10.6084/m9.figshare.9944597.v3`.



MODE

Right: scheme of the modeled apparatus (graph courtesy **G. C. Strong**)

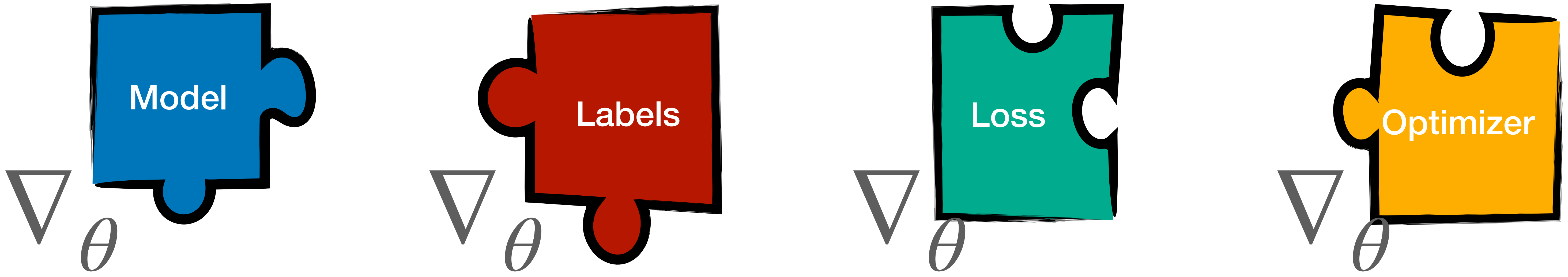


[Strong, Dorigo]

Differentiable Programming as a paradigm

HEP & ML are a great match - slowly permeating everything

Gradient Information allows us to inject physics domain knowledge into ML and make them more data-efficient, interpretable and robust systems



add inductive bias (symmetries, structure…) | exploit simulators | judge on physics goals | (left for homework)

1990

Making the World Differentiable: On Using Self-Supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments

Jürgen Schmidhuber*
Institut für Informatik
Technische Universität München
Arcisstr. 21, 8000 München 2, Germany

2022

Differentiable Programming in High-Energy Physics

Atılım Güneş Baydin (Oxford), Kyle Cranmer (NYU), Matthew Feickert (UIUC), Lindsey Gray (FermiLab), Lukas Heinrich (CERN), Alexander Held (NYU) Andrew Melo (Vanderbilt) Mark Neubauer (UIUC), Jannicke Pearkes (Stanford), Nathan Simpson (Lund), Nick Smith
Savannah Thais (Princeton), Vassil Vassilev

August

Abst

A key component to the success of deep learnin
learning practitioners compose a variety of modules

Differentiable Simulators for HEP

L. Heinrich[1], M. Kagan[*2], M. Mooney[3], and K. Terao[2]

[1]CERN