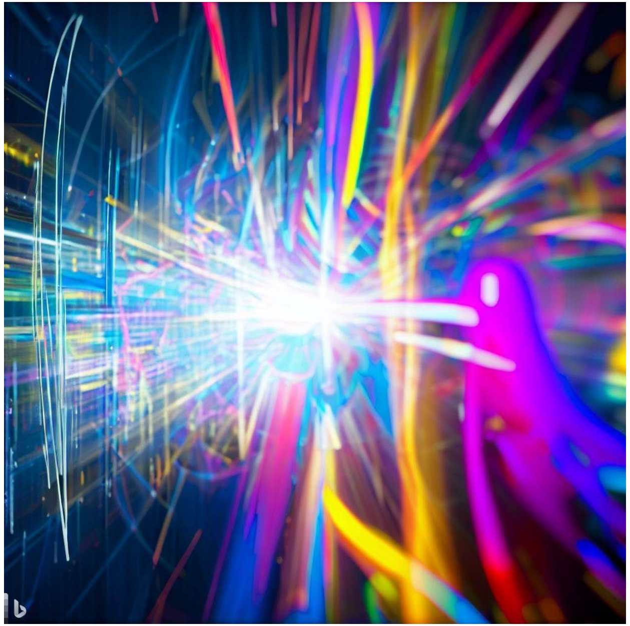
XX Seminar on Software for Nuclear, Subnuclear and Applied Physics
4-9 June 2023, Hotel Porto Conte, Maristella SS, IT



Machine Learning in Particle Physics

Lucio Anderlini INFN Firenze

Istituto Nazionale di Fisica Nucleare
SEZIONE DI FIRENZE

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

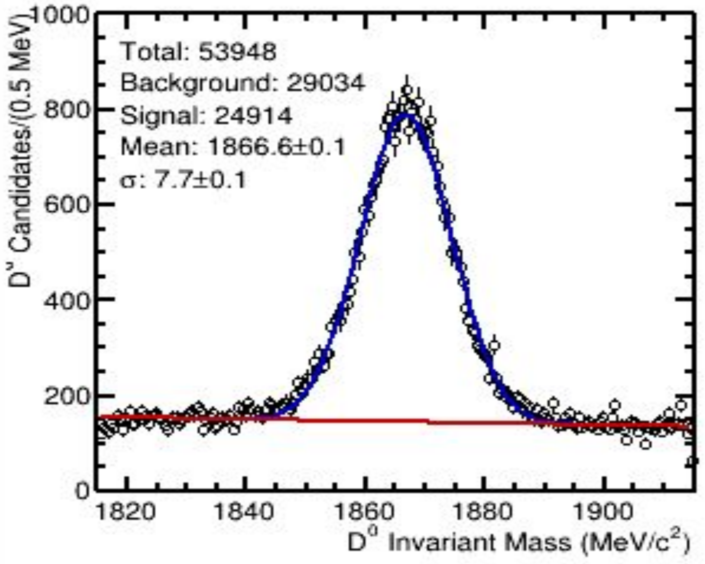
Unbinned Maximum Likelihood Fits as ML algorithms

The maximum likelihood principle is widely used in physics to model datasets.

The likelihood is defined as

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} f(x_i; \boldsymbol{\theta})$$

where the x_i represent the dataset, and $\boldsymbol{\theta}$ is a vector of **parameters** the probability density function f depends on.

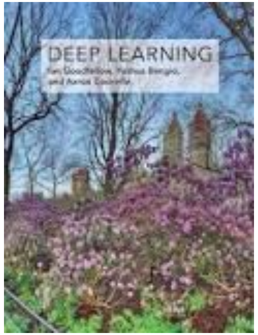


If you are interested in the values on $\boldsymbol{\theta}$ then it's **parametric density estimation.**

If $\boldsymbol{\theta}$ is just needed to define a shape, then it's **non-parametric density estimation**, a widely investigated research topic in machine learning.

* Picture from the LHC Masterclasses programme, for the LHCb experiment

Convergence. Machine Learning *is* fitting



"Most modern neural networks are trained using <u>maximum likelihood</u>. This means that the cost function is simply the negative log-likelihood."

I. Goodfellow, Y. Bengio and A. Courville, *"Deep Learning"*, MIT Press (2016)

zfit provides model building and fitting on top of TensorFlow.

www.tensorflow.org ▾ Traduci questa pagina

TensorFlow

An end-to-end open source machine learning platform.

zfit: scalable pythonic fitting

Jonas Eschle, Albert Puig Navarro, Rafael Silva Coutinho, Nicola Serra

Physik-Institut, Universität Zürich, Zürich (Switzerland)

https://arxiv.org/abs/1910.13429

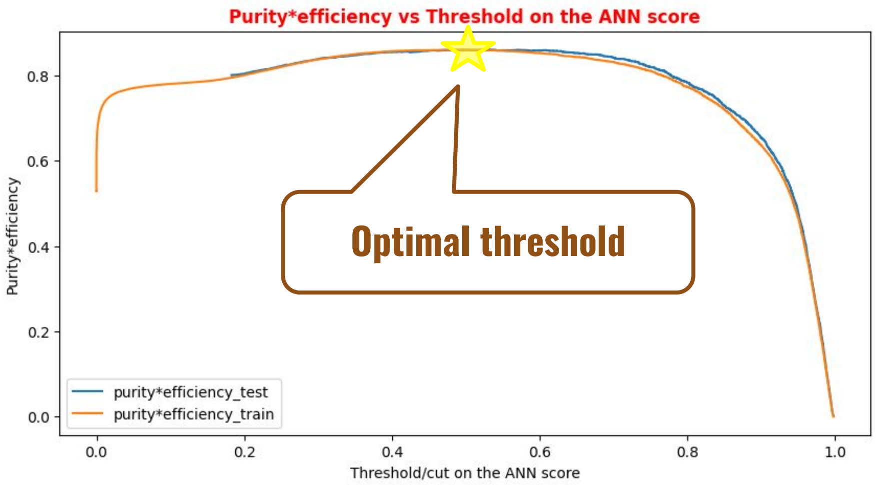
Abstract

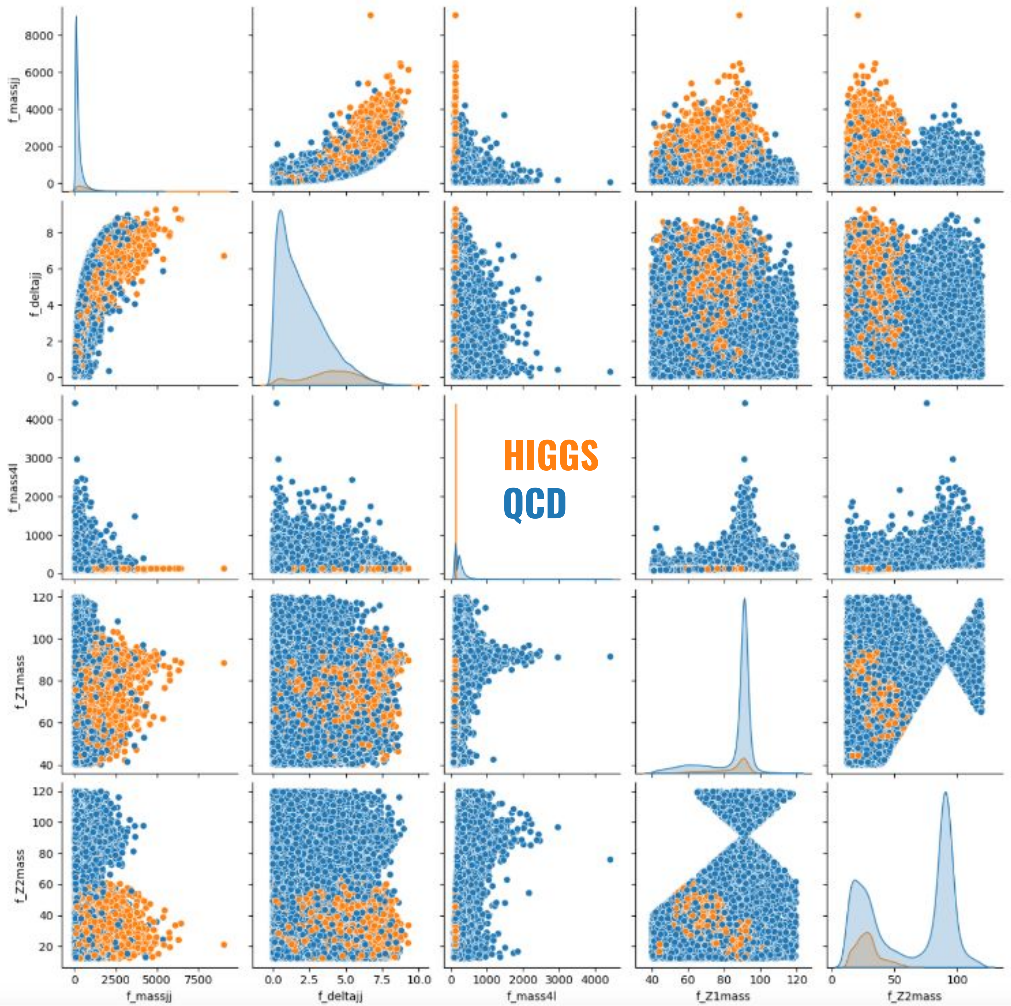
Statistical modeling is a key element for High-Energy Physics (HEP) analysis. The standard framework to perform this task is the C++ ROOT/RooFit toolkit; with Python bindings that are only loosely integrated into the scientific Python ecosystem. In this paper, zfit, a new alternative to RooFit written in pure Python, is presented. Most of all, zfit provides a well defined high level API and workflow for advanced model building and fitting together with an implementation on top of TensorFlow. It is designed to be extendable in a very simple fashion, allowing the usage of cutting-edge developments from the scientific Python ecosystem in a transparent way. Moreover, the main features of zfit are introduced, and its extension to data analysis, especially in the context of HEP experiments, is discussed.

Classification

Multivariate analyses have been used in Particle Physics for many years to classify signal events from background.

We often design the whole analysis workflow to **provide data-driven samples**.





Classification

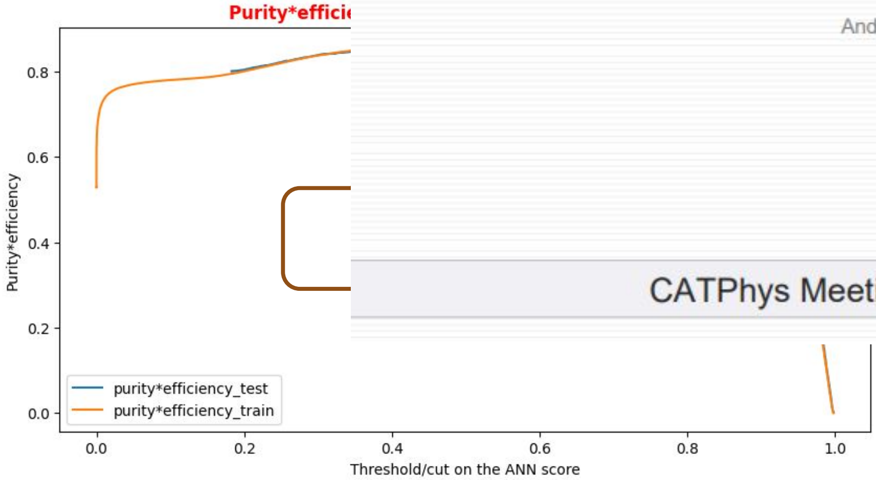
Multivariate ana

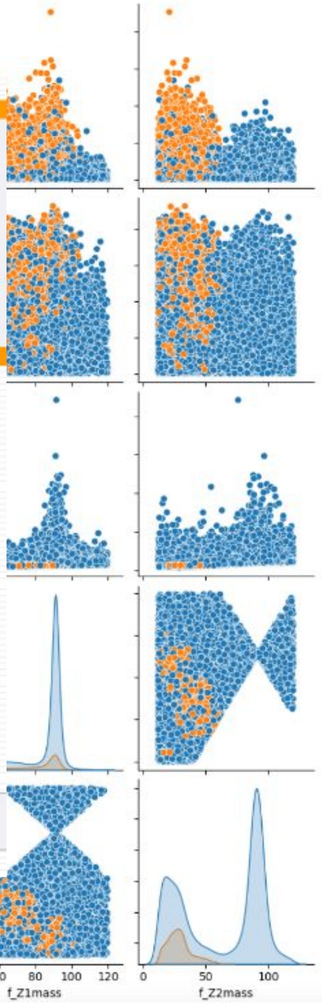
Particle Physics f

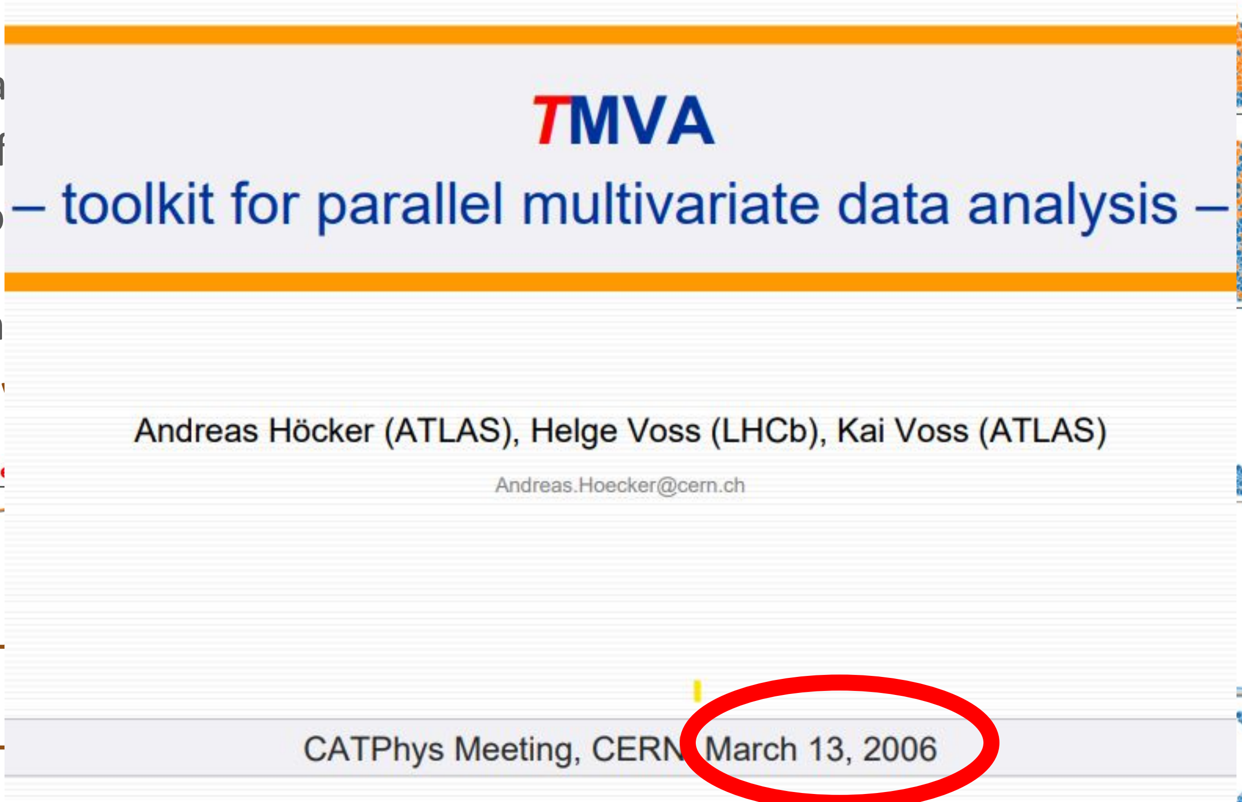
signal events fro

We often design

workflow to **pro**



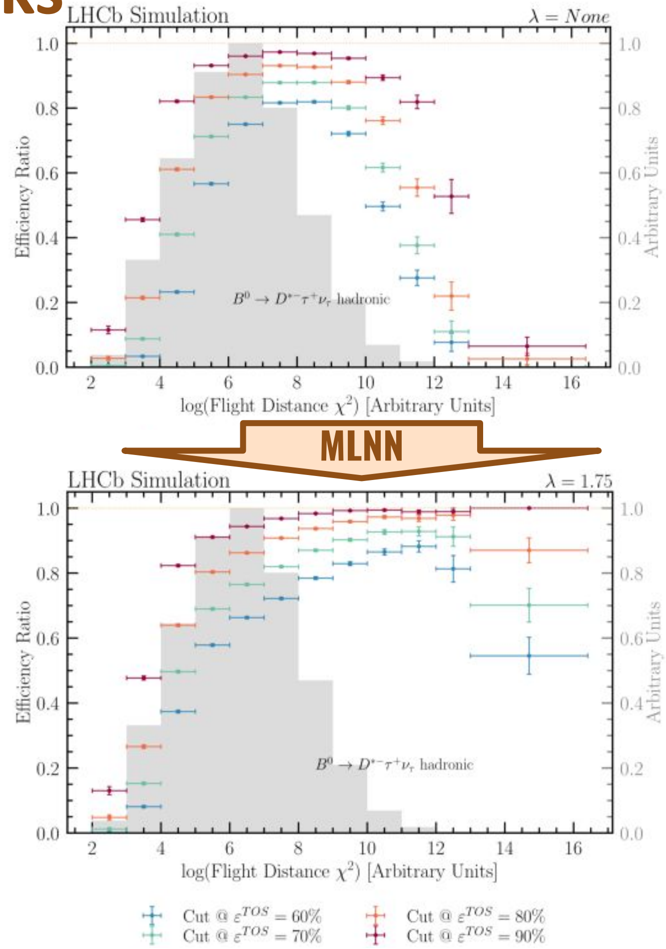




Monotonic Lipschitz Neural Networks

Development on techniques to improve our capabilities in the classification task are still very active.

In December 2021, an LHCb-team proposed a new class of neural networks targeting the exact same problem.

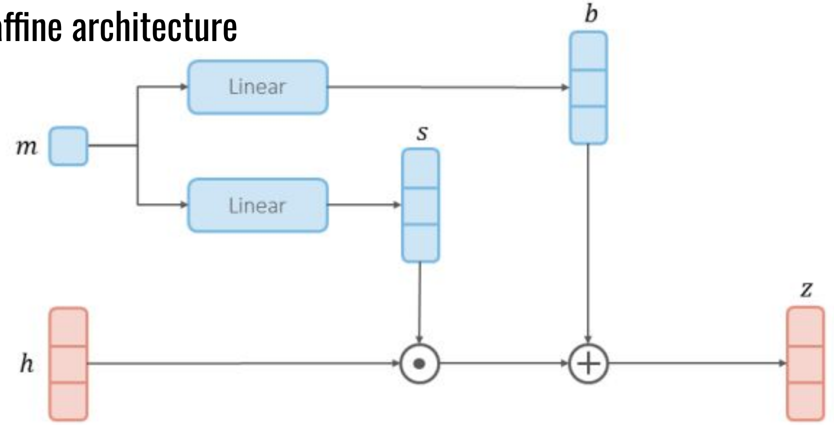


Monotonic Lipschitz Neural Networks

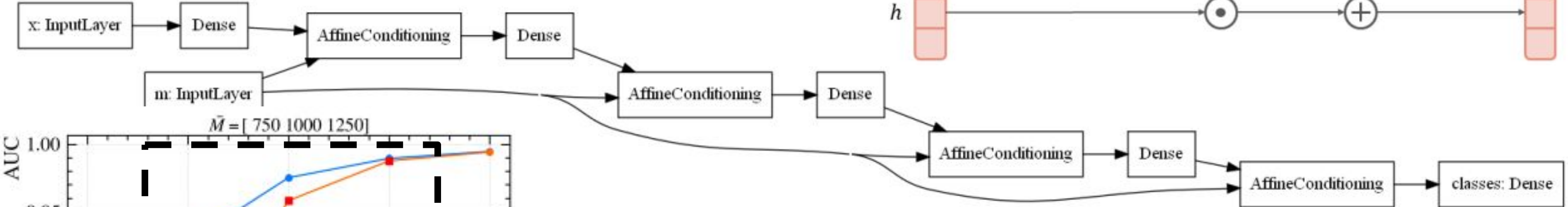
- **Robust** against small changes (*e.g.* experimental instabilities, data/MC discrepancies)
- **Monotone** with respect to (at least) certain features: *e.g.* the higher the transverse momentum the better

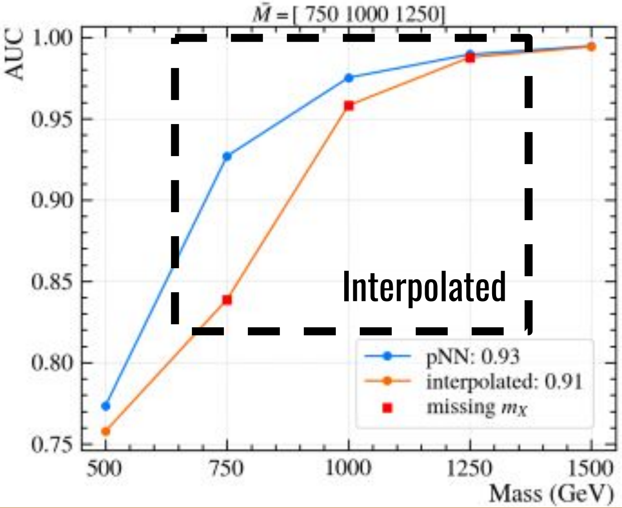
Parametric Classification

The affine architecture



Loosening the assumptions on at least one of the samples (for example the signal) is another active area of research.





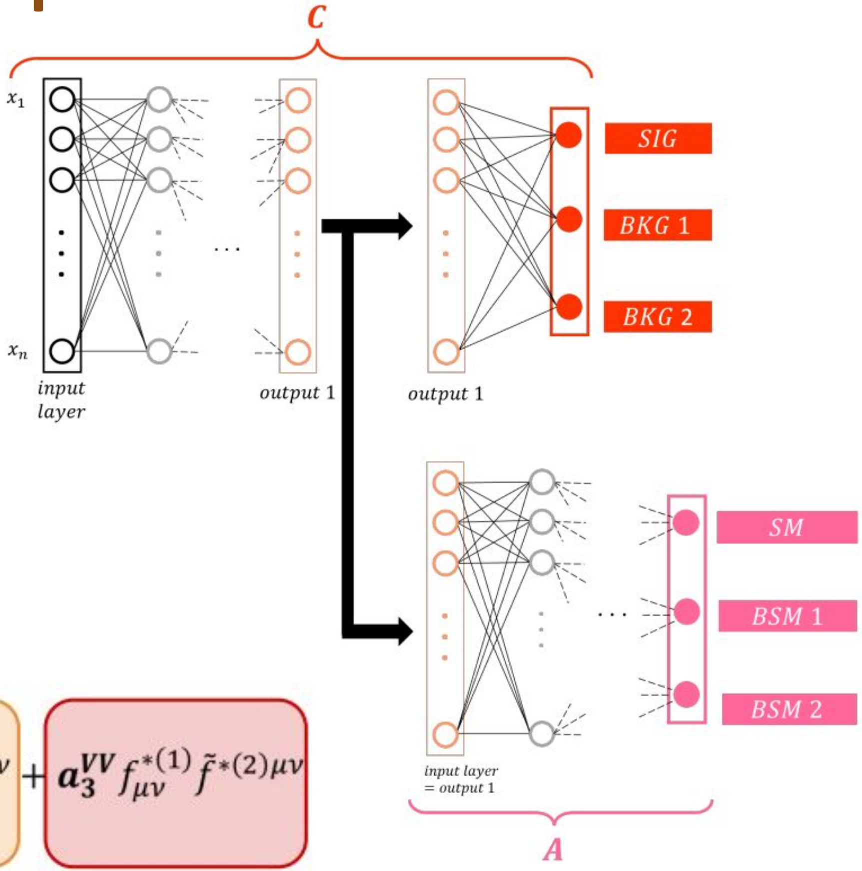
Parametric Neural networks enable training classifiers robust against one or more parameters defining, for example, the signal distribution.

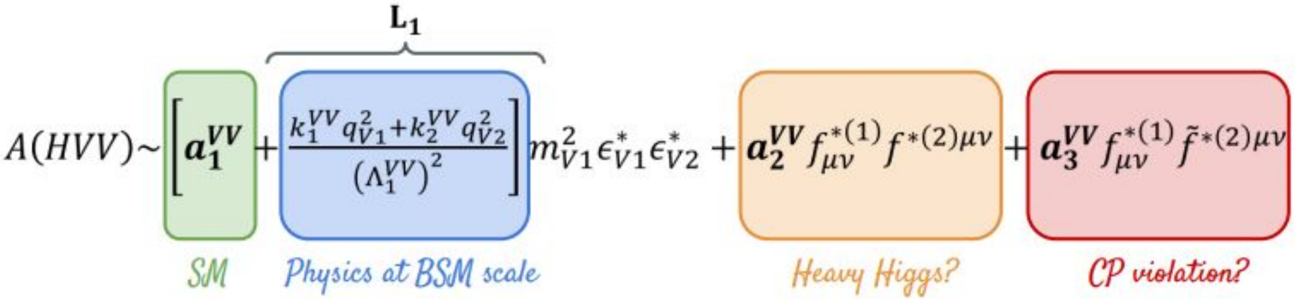
Not only this leads to **better performance**, but it enables interpolating to get **classifiers for unforseen hypotheses**.

Classification with domain adaptation

An alternative approach is to use adversarial training to train networks that actively ignore some feature of the input distributions.

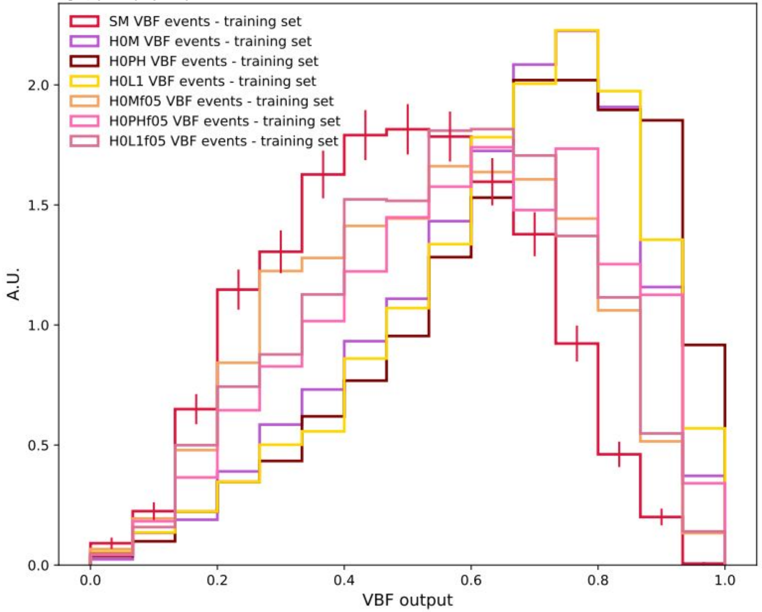
For example, one can train a classifier to **actively ignore differences in the signal distributions due to contribution Beyond the Standard Model**
→ *Model independent searches!*



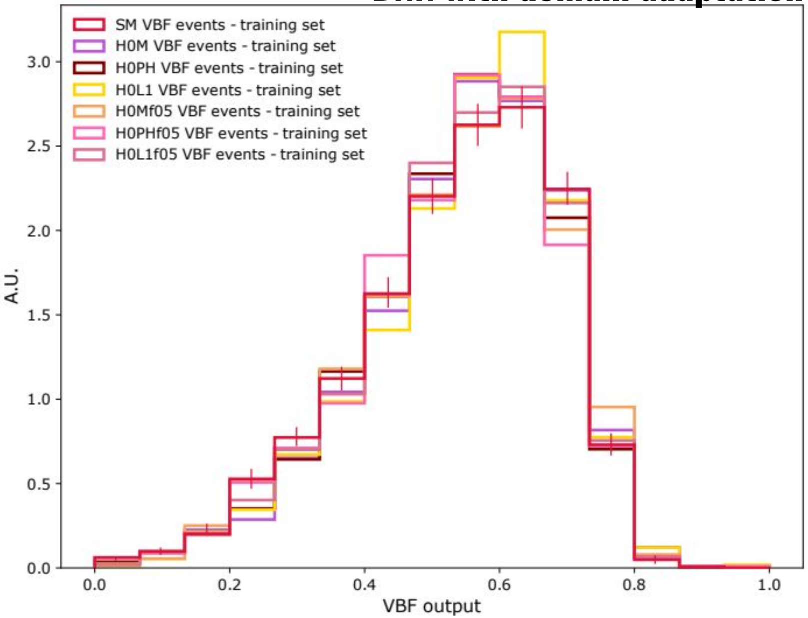


Classification with domain adaptation

Standard DNN



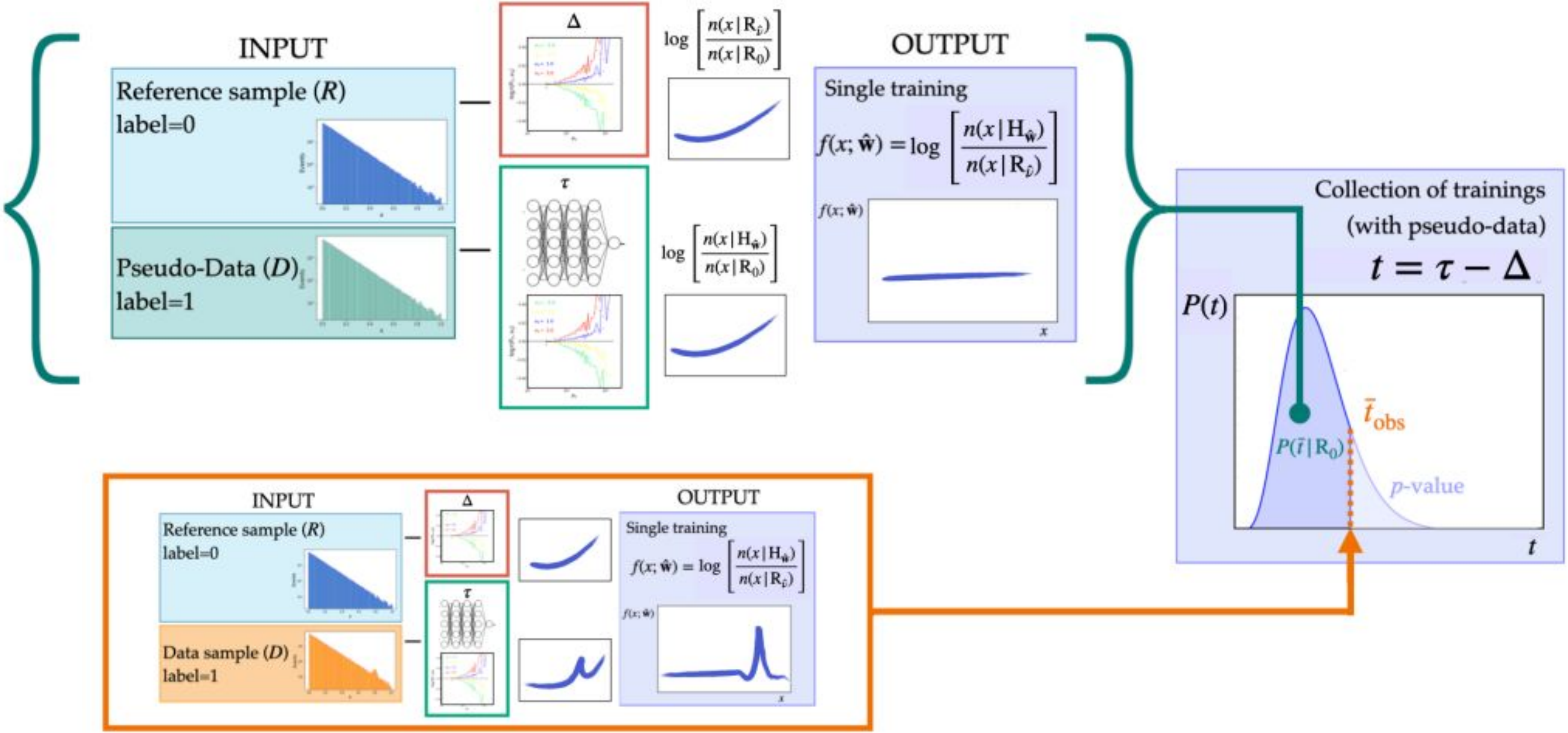
DNN with domain adaptation



C

SM Physics at BSM scale Heavy Higgs? CP violation?

Classifying the unexpected: *Anomaly Detection*



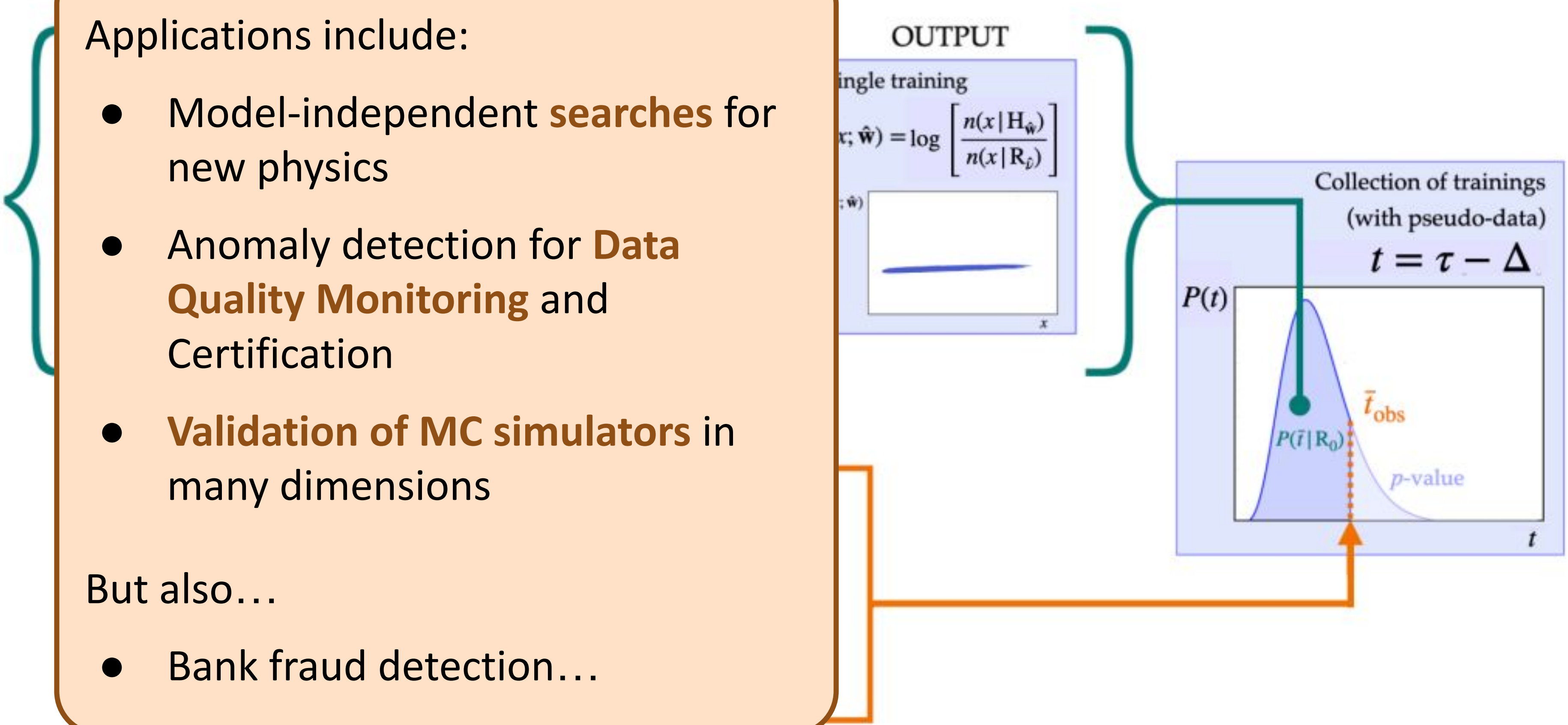
Classifying the unexpected: *Anomaly Detection*

Applications include:

- Model-independent **searches** for new physics
- Anomaly detection for **Data Quality Monitoring** and Certification
- **Validation of MC simulators** in many dimensions

But also…

- Bank fraud detection…



ML Classification in the hardware trigger

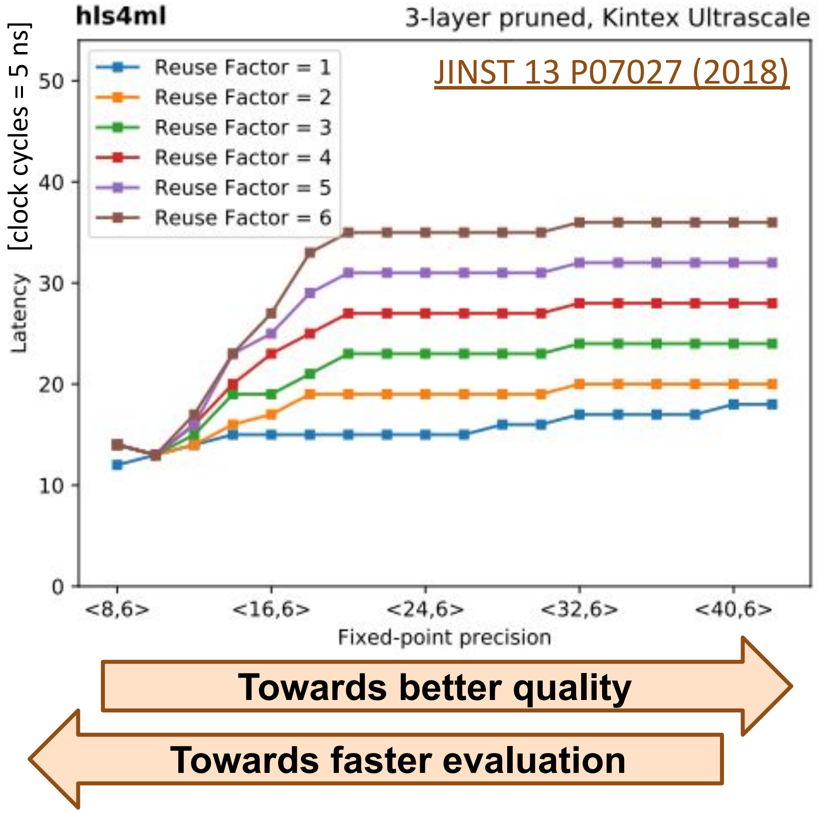
a.k.a. online selection
→ Before storing data!

Move classification algorithms to hardware to make them:

- **faster** (and less energy-consuming)
- running at **fixed latency**

Deployment in hardware enables **tuning performance**, for example:

- increasing the representation error of real numbers (**fixed-point algebra**)
- **pruning nodes**, skipping unnecessary multiplications)



Deployment on hardware, two approaches

Using *High Level Synthesis* languages

1. Translate your model to C
2. Use **"special compilers"** to deploy compiled C on FPGA (programmable hw devices)

JINST 13 P07027

Industry standard: *fast and reliable*

Industry standard: *commercial licenses and potential vendor lock-in*

High Level Synthesis (HLS) is a design methodology that enables the generation of synthesizable hardware designs from high-level descriptions of functionality. HLS can be used to **accelerate applications on FPGAs** by using C/C++ as the design entry and **reducing the development time and effort**.

Embedding specialized μprocessors in HW

Embed **custom "micro-processors"** with custom instruction set to the FPGA.
Code the NN (or any other algorithm) for those μP, compiling in **assembly-like**

Par. Comp. 109 102873

Research product: *full control of the software stack (open source)*

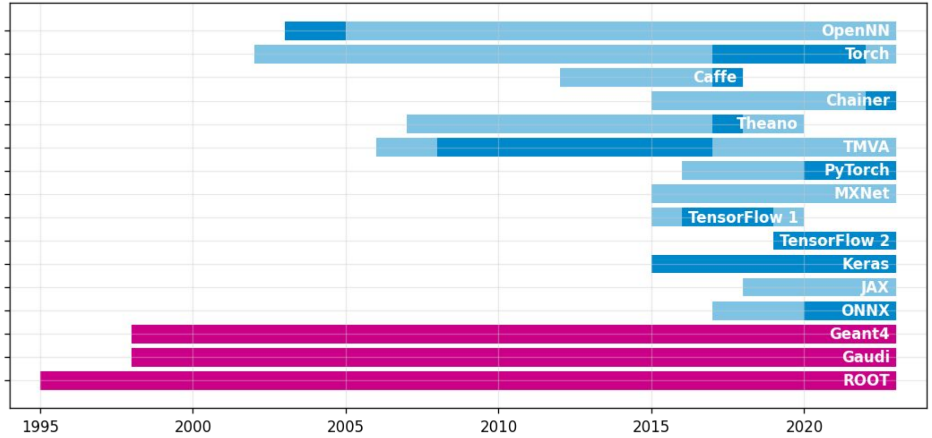
Research product: *will always remain "under active development"*

Deployment on software (trigger)

Multiple options are available and are being explored for deploying ML in HEP applications (C++ distributed on WLCG).

- *project lifecycle;*
- *throughput;*
- *single-call overhead*

should all be considered for an optimal selection



Model complexity & Throughput

Accelerated Services
MLaaS4HEP
FaaST

Dedicated Runtimes
Tensorflow
C/C++ TorchLib
ONNX

C++ libraries
SOFIE/TMVA
LWTNN

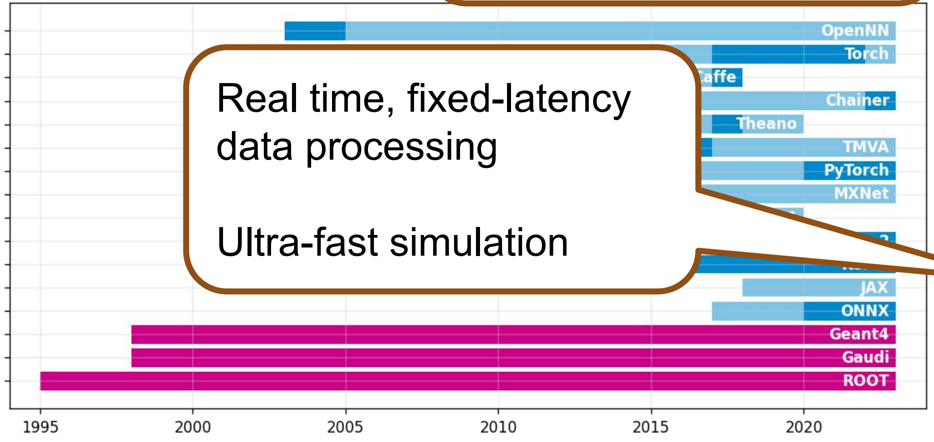
C transpilers
keras2c
scikinC

Overhead on single-evaluation

Deployment on software (trigger)

Multiple options are available and ar
being explored for deploying ML in H
applications (C++ distributed on WLCG).

- *project lifecycl*
- *throughput;*
- *single-call over*



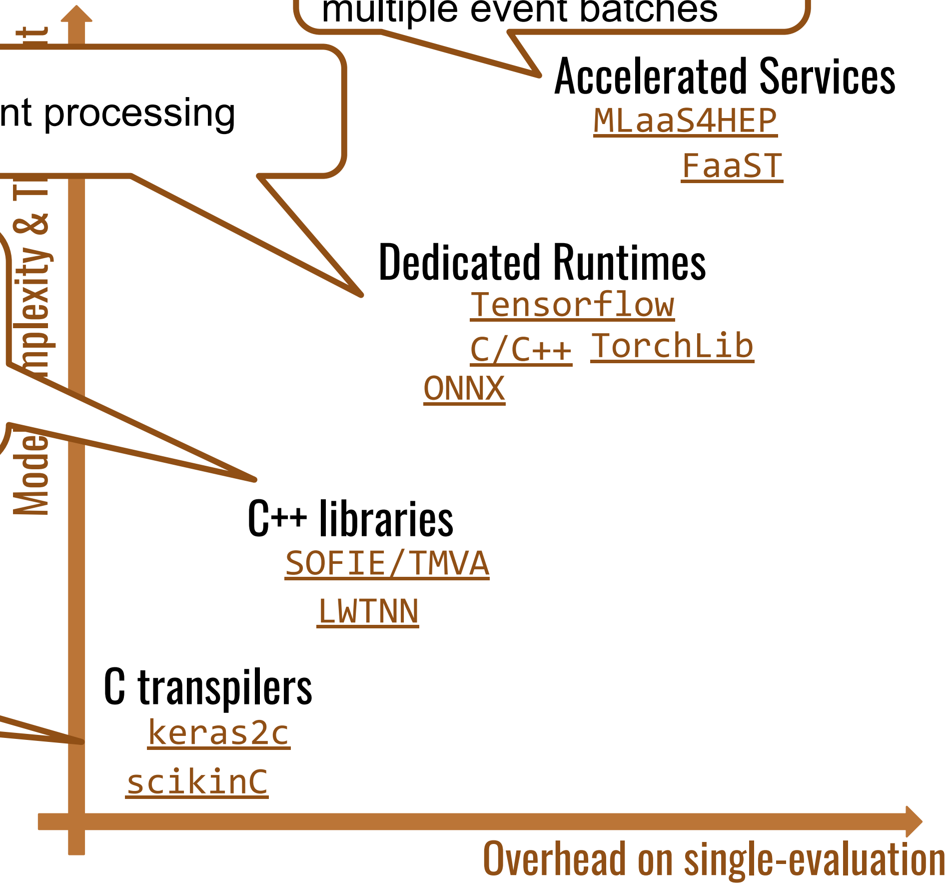
Real time, fixed-latency data processing

Ultra-fast simulation

Reconstruction-level classifiers and regressors on single objects

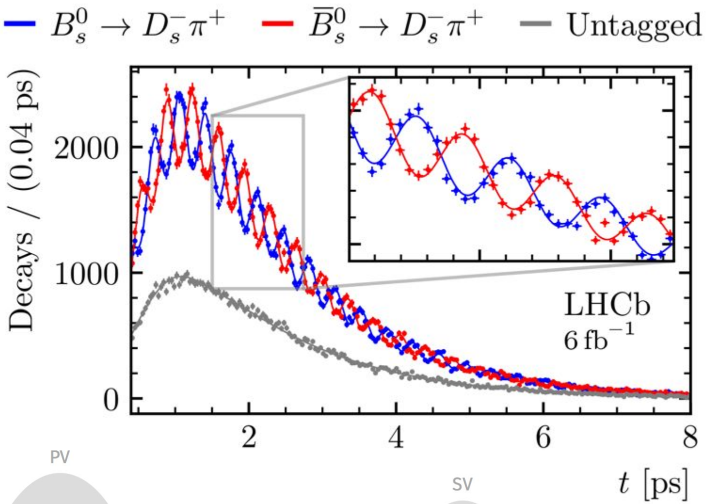
Full-event processing

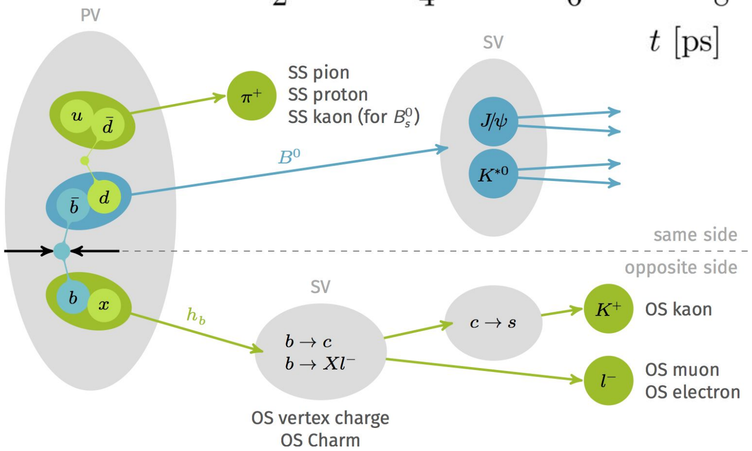
Full-event processing on very large models; multiple event batches



Upcoming future

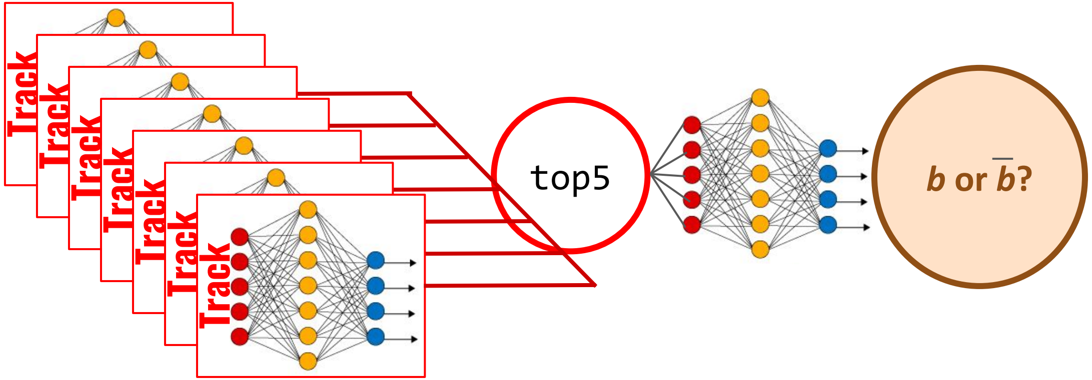
Flavour Tagging with custom architectures





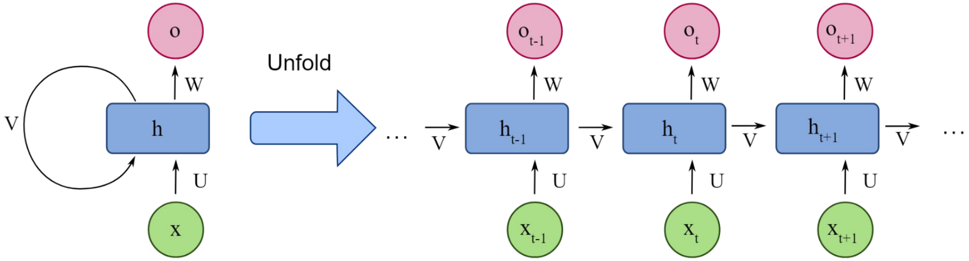
Neutral b mesons are very peculiar in that they can **oscillate** while they fly. To study this phenomenon one needs to tag the flavour of the b meson **when it was produced**, and compare it to the flavour when it decayed.

Clever architectures were used already in ante-TensorFlow era to interpret "**the other particles**"



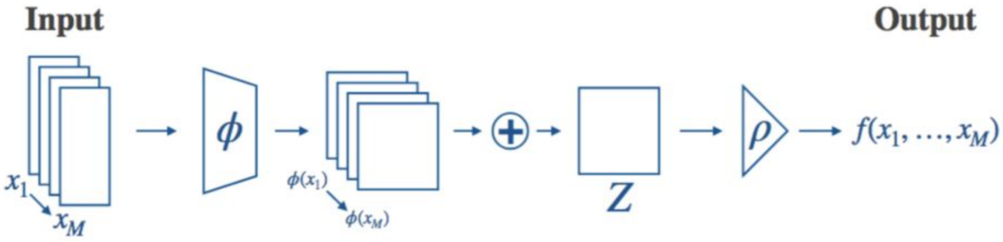
Flavour tagging with RNNs and DeepSets

Recurrent Neural Networks enabled to extend significantly the number of “other particles” processed, increasing the tagging performance.

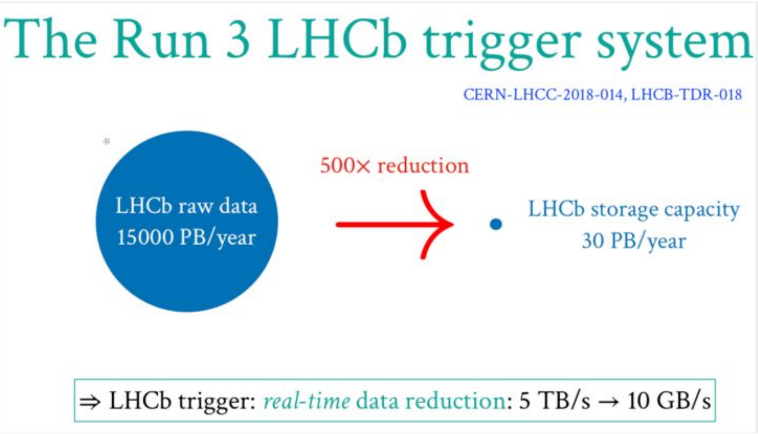


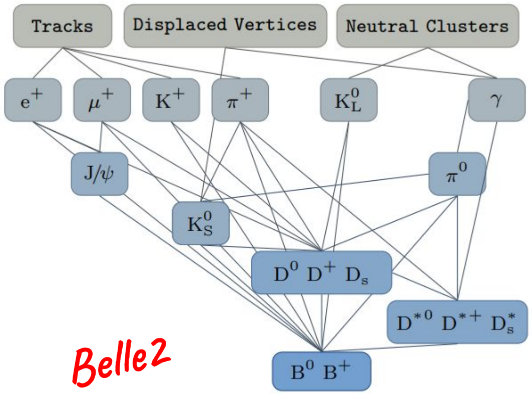
Unfortunately they are slow.

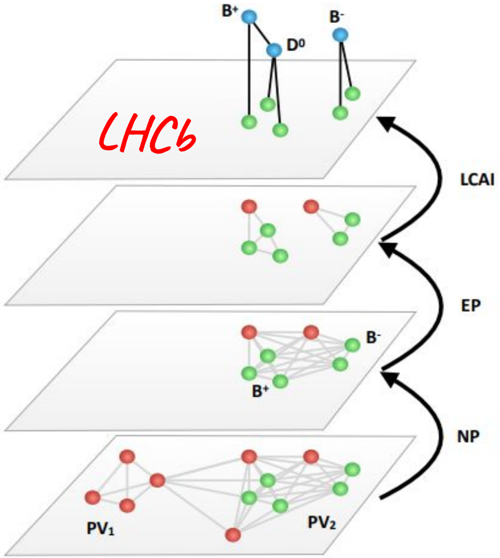
Deep Sets (which are a special case of Graph Neural Networks) were shown to outperform RNNs in terms of speed, with the same tagging performance



Global-Event Interpretation in the Software trigger





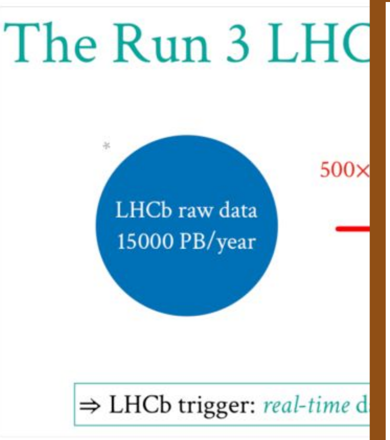


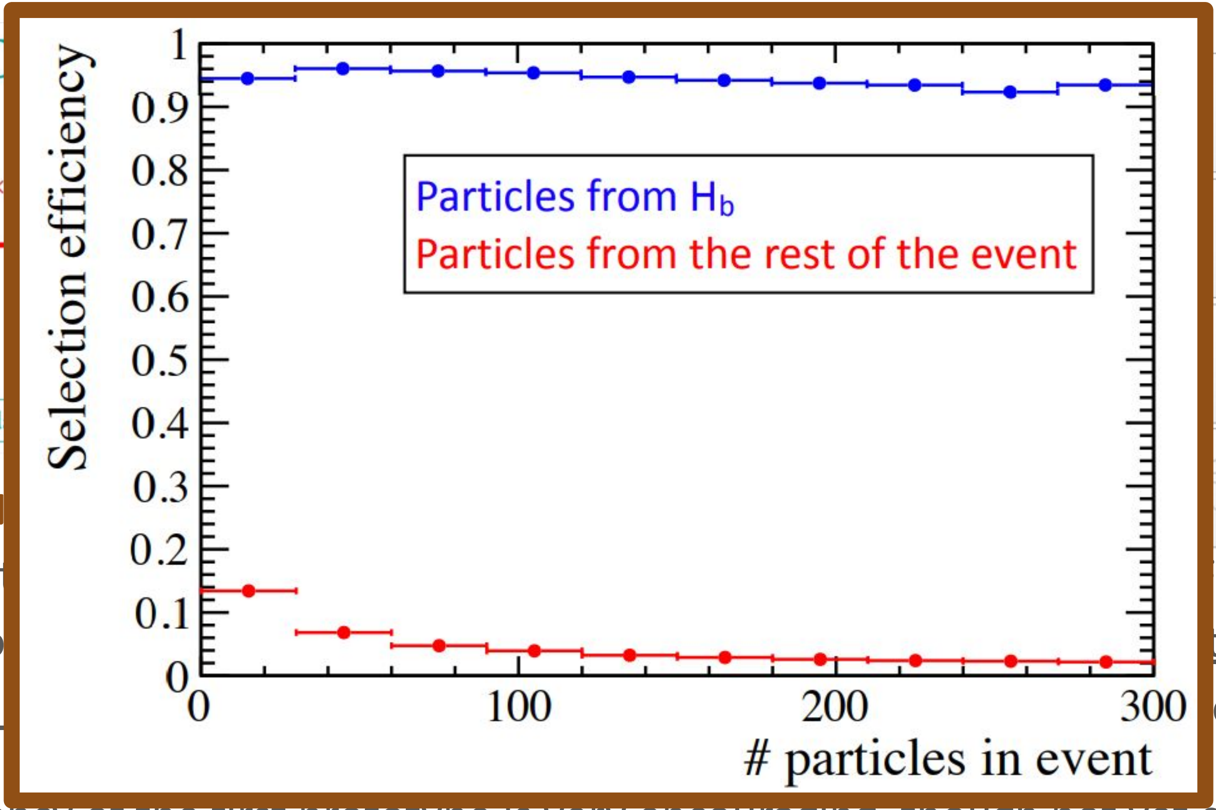
Graph Neural Networks are being studied also to interpret the whole event in one go: **Deep-learning-based Full Event Interpretation** (DFEI)

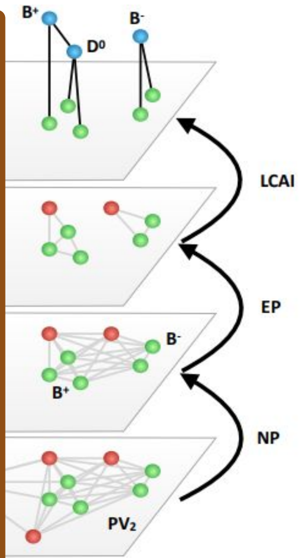
This is crucial for future b-physics experiments at hadronic machines as <u>almost all events will contain a *b* quark</u>: triggering will mean ***"select a part of the event"*** to store offline.

Selection efficiency of the first prototype is very encouraging, though not yet competitive with human-tuned, single-decay-mode selection strategies.

Global-Event Interpretation in the Software trigger







Graph Neural N

the whole event … EI)

This is crucial fo … all events

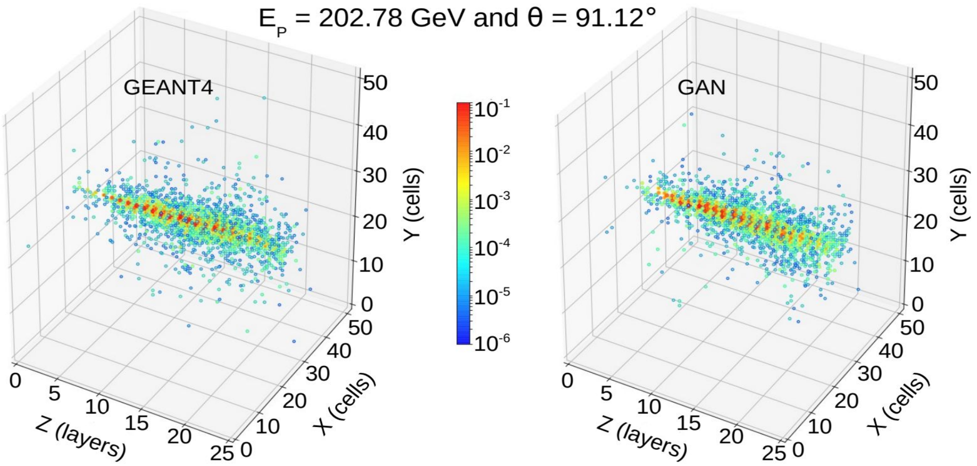
will contain a *b* … e offline.

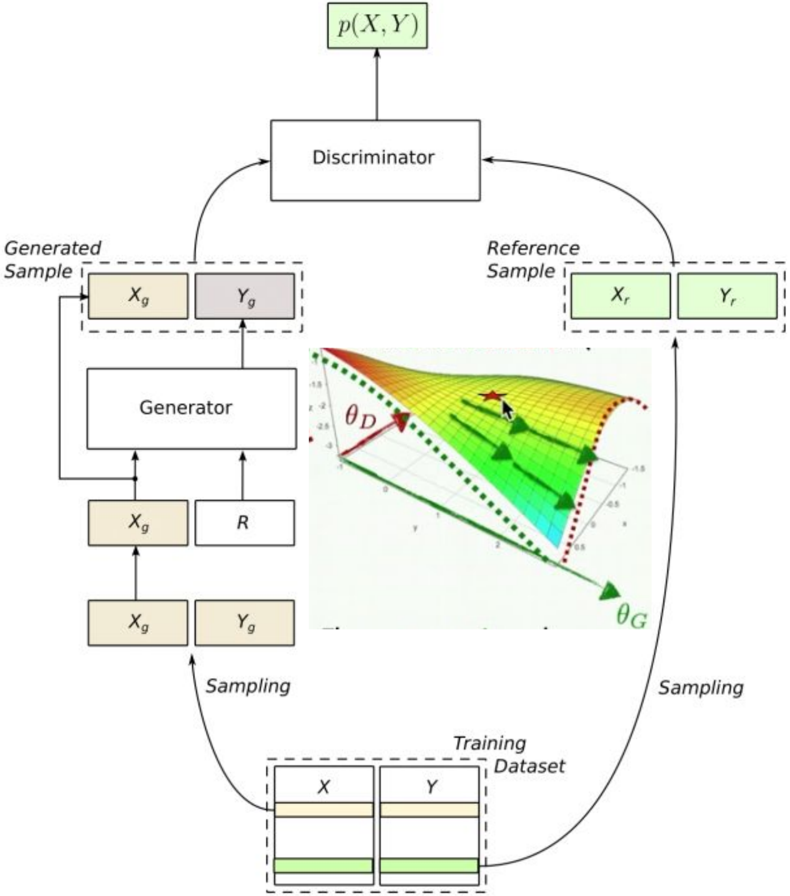
Selection efficiency of the first prototype is very encouraging, though not yet competitive with human-tuned, single-decay-mode selection strategies.

Generative models for detector simulation

Deep Generative Models can also drastically speed-up the detector simulation.

Instead of computing a shower for each particle hitting a Calorimeter, we can **statistically model it**

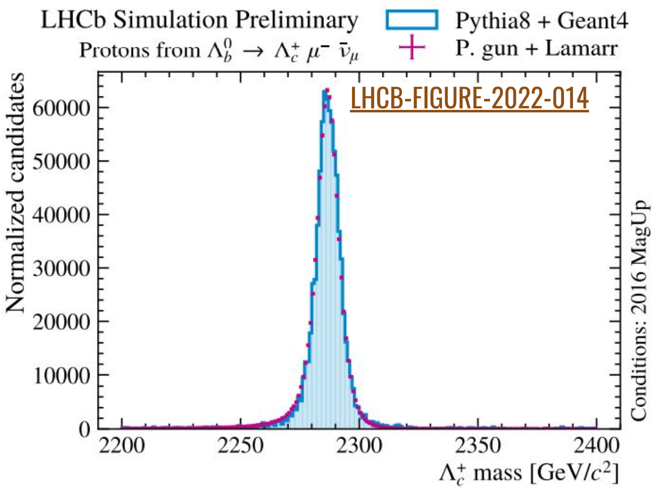


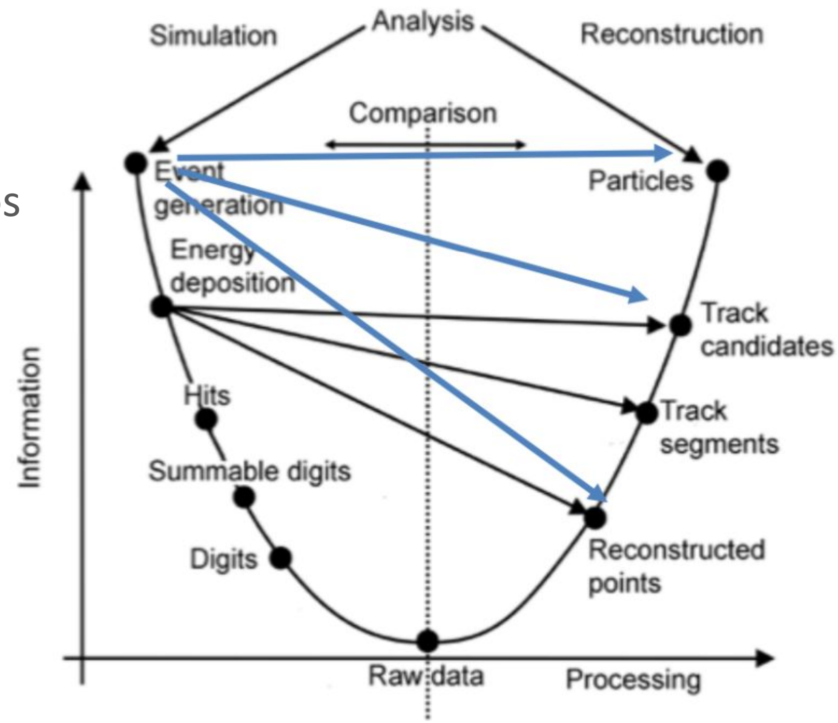


Ultra-fast (or Flash) simulation

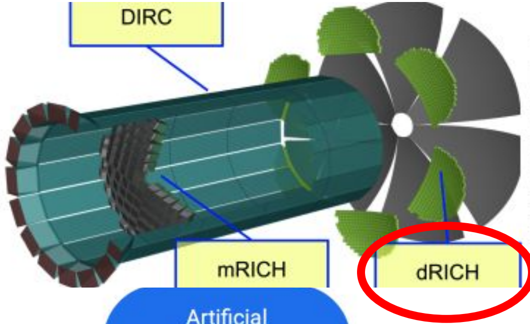
We can be even more aggressive and parametrize both the detector simulation and the reconstruction algorithms.

With "**ultra-fast simulation**" we can achieve speed-ups O(1000×) with respect to Geant4-based simulation

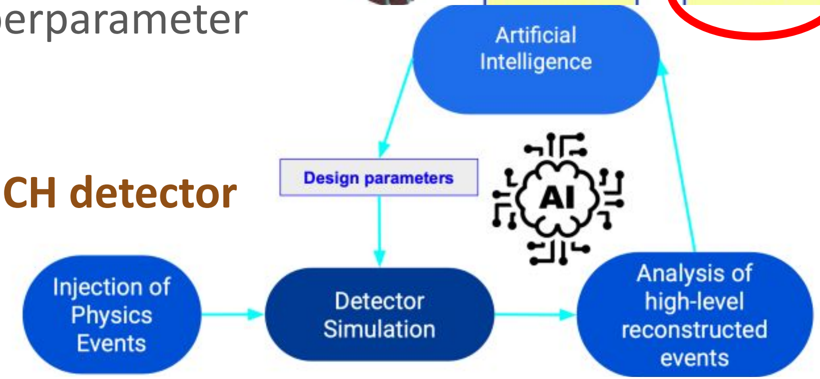




Digital Twins and Bayesian Optimization



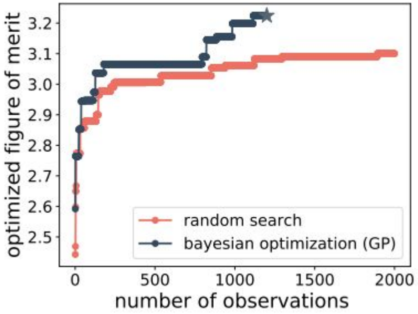
Simulation (and fast simulation) can then be exploited to automate the detector optimization procedure, simulating **AI-chosen detector options** (as you would do for Hyperparameter Optimization).

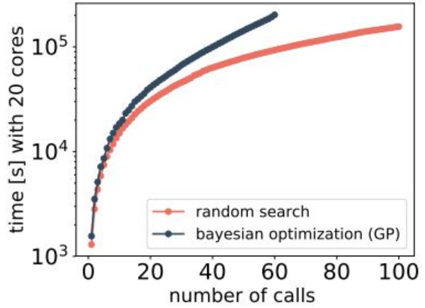


For example, **the geometry of the double-radiator RICH detector** at **EIC was tuned using Bayesian Optimization**.

8 parameters

description	range [units]
mirror radius	[290, 300] [cm]
radial position of mirror center	[125, 140] [cm]
longitudinal position of mirror center	[−305, −295] [cm]
shift along x of tiles center	[−5, 5] [cm]
shift along y of tiles center	[−5, 5] [cm]
shift along z of tiles center	[−105, −95] [cm]
aerogel refractive index	[1.015, 1.030]
aerogel thickness	[3.0, 6.0] [cm]



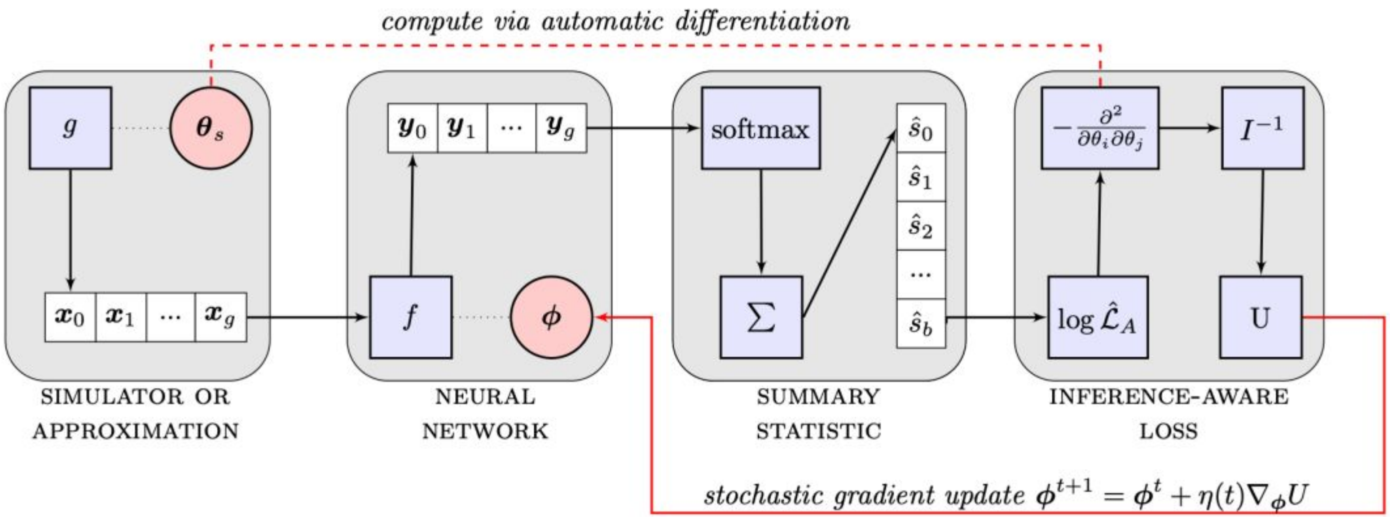


Further future
(and possibly a bit speculative)
applications

Parametric programming for Detector Optimization

What if we could use stochastic gradient descent to optimize the detector parameters themselves?

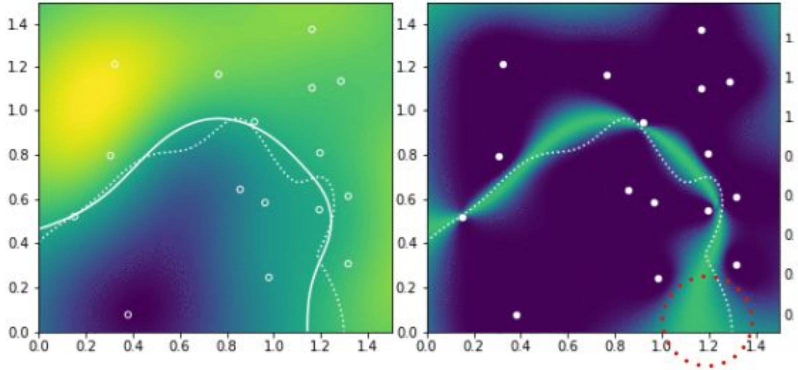
Clearly, this would require **rewriting everything** from simulation to analysis with differentiable programming techniques.



Active learning

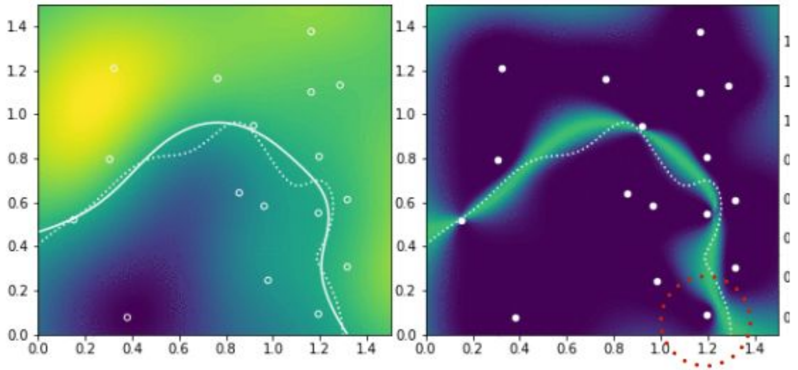
Also, instead of tuning the detector parameters we may leave the Bayesian Estimation to **tune the theoretical models for which we need simulated samples**, to reduce the number of simulated events where they do really matter.

This interplay between analysis and experiment, where the AI decides which experiment to perform, is named **active learning** and we are just beginning to appreciate its power.



from L. Heinrich's talk at the 4th reinterpretation workshop

lots of uncertainty in contour here



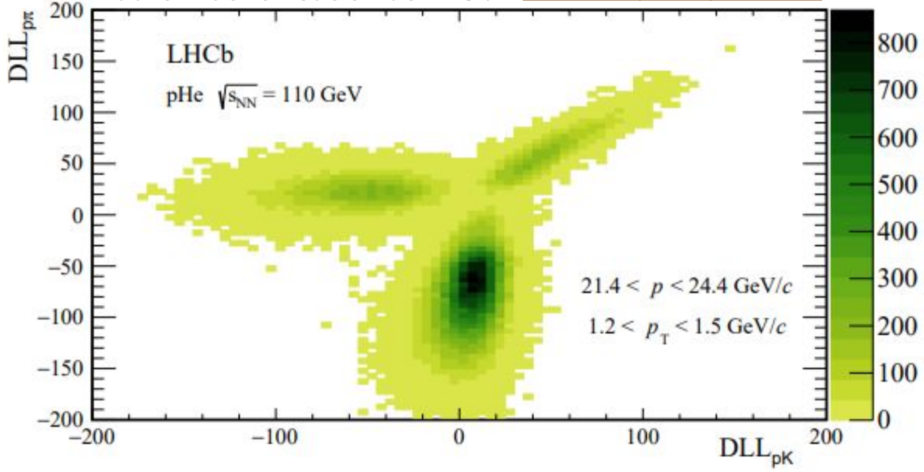
high-value point close to contour

Probabilistic reconstruction

We could **extend our reconstructed objects** with "probabilistic" information.

We already do for covariance matrices for tracking (d=6), we do for hadron identification (d=3), can we do a "d=1000"?

Hadron identification at LHCb PRL 121 (2018) 222001



Imagine that like storing the inter**nal representation of a DNN layer in the reconstructed objects** (e.g. a track), storing probabilistic information on how the track was identified and reconstructed.

Could tune them by building the reco sequence with differentiable programming.

Is that worth the additional storage? And the additional computing effort?
How would we treat that at analysis-level?

Large Language Models for code porting and…

LLMs are the hype these days.

Can we, ethically, benefit from language models?

- ~~Writing grant proposals?~~
- ~~Writing proceedings?~~
- ~~Preparing slides?~~
- Porting code from the current huge code-base for few-threaded processes to HPC and HW-accelerated environments?

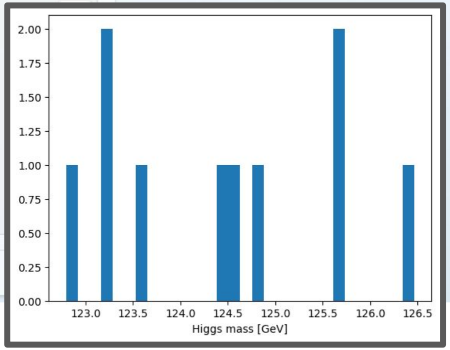
Write a python code to download csv data from https://opendata.cern.ch/record/5209/files/diphoton.csv and plot the histogram of the M variable. Title the axis "Higgs mass [GeV]".

Here is a Python code that you can use to download the csv data from the given URL and plot the histogram of the M variable with the specified axis title:

```
import pandas as pd
import matplotlib.pyplot as plt

url = 'https://opendata.cern.ch/record/5209/files/diphoton.csv'
data = pd.read_csv(url)

plt.hist(data['M'], bins=30)
plt.xlabel('Higgs mass [GeV]')
plt.show()
```



If you have ideas, ~~let ChatGPT generate and~~ submit an abstract to the [1st Large Language Models in Physics Symposium (LIPS)]

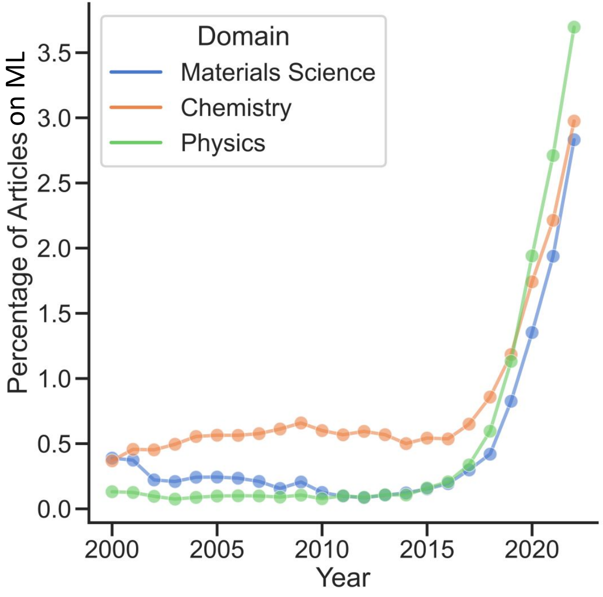
Conclusion

Machine Learning and Artificial Intelligence have been a world-wide hype for 7 years now.

And it's changing Science.

HEP has robust statistical foundations and traditions, and a long history of applying MVA algorithms to a wide variety of applications. **And much has to come**.

It is important to realize, however, that Science (and especially **Fundamental Science**) is not about making things statistically work in most cases, but aims at **Scientific Understanding**.



That's the only hard limit to ML applications to HEP: we deal with digital data since the '90s

Acknowledgements

Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

This work is partially supported by ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU.

Finanziato dall'Unione europea NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing