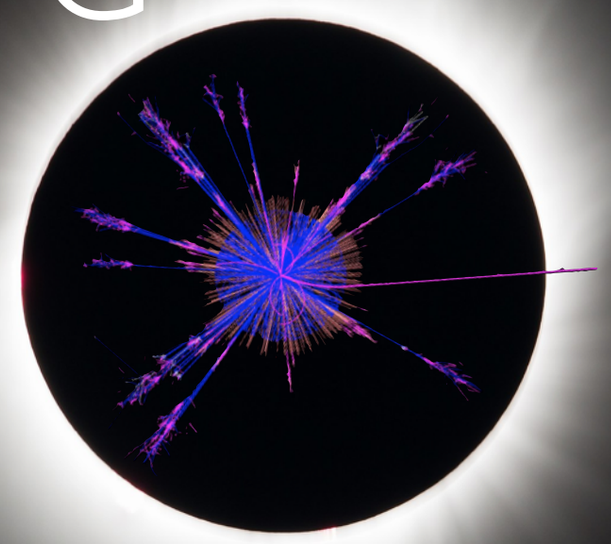
NYU Center for Data Science

Center for Cosmology and Particle Physics

THE BLEEDING EDGE OF

PHYSICS-AWARE MACHINE LEARNING



@KyleCranmer
New York University
Department of Physics
Center for Data Science
CILVR Lab

TOPICS

Evolution of the use of machine learning in physics:

- **traditional:** classification & regression
- **emerging:** inference & generation

Reorganization of the high-level HEP workflow

- active learning for more efficient use of resources
- containers & reusable workflows

Impact of these on HEP software & computing

THEME: GET CLOSER TO RAW DATA

Exploring end-to-end deep learning solutions for event classification at CMS

Michael Andrews[1,2], Manfred Paulini[1,2], Sergei Gleyzer[1,3], Barnabas Poczos[4]
on behalf of the CMS experiment
[1]CMS, [2]Carnegie Mellon University-Physics, [3]University of Florida, [4]Carnegie Mellon University-ML

ACAT 2017, 2017-AUG-22

Physics Carnegie Mellon

CMS

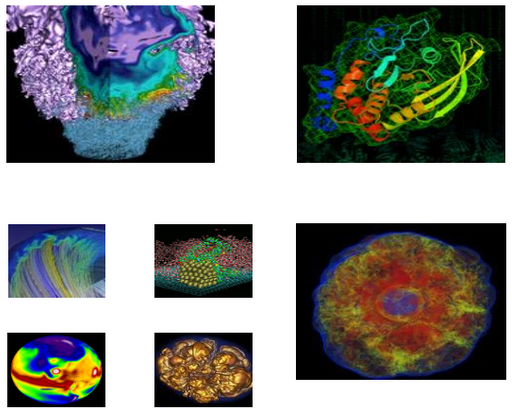
Deep learning in jet reconstruction at CMS

This Report is part of a project that has received funding from the **European Union's Horizon 2020 research and innovation programme under grant agreement N°675440**

Markus Stoye[1,2] for the CMS collaboration
CERN[1], ITN aMVA4newphysics[2]

ACAT, 22nd August 2017

Deep Neural Networks for Physics Analysis on low-level whole-detector data at the LHC



Wahid Bhimji, Steve Farrell, Thorsten Kurth, Michela Paganini, Prabhat, Evan Racah
Lawrence Berkeley National Laboratory
ACAT 2017: 21st August 2017

NERSC 40 YEARS at the FOREFRONT

U.S. DEPARTMENT OF ENERGY | Office of Science

- 1 -

BERKELEY LAB

ATLAS EXPERIMENT

Yale

MACHINE LEARNING FOR B-JET TAGGING

MICHELA PAGANINI
ON BEHALF OF THE ATLAS COLLABORATION

THEME: IMAGES & END-TO-END LEARNING

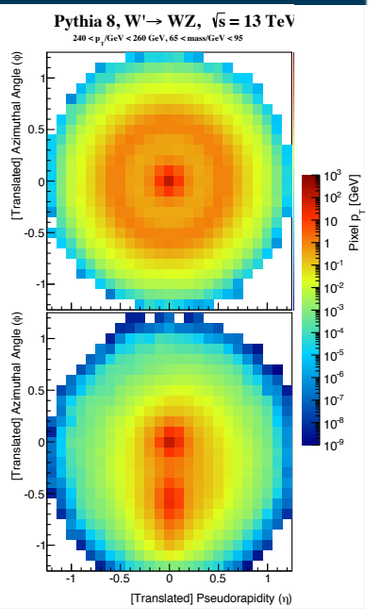
Pre-processing and Special Relativity

9

Pre-processing is an important aspect of image recognition

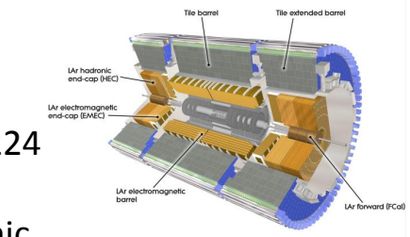
However, some steps can ***damage*** *the physics information content* of a physics image

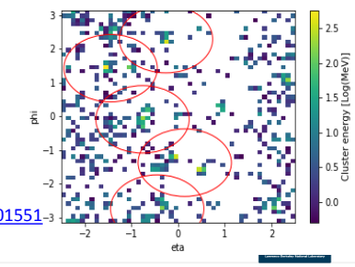
I won't discuss this in detail here, but I bring it up so you are aware of it!



Data processing

- **Bin calorimeter tower energy in η/ϕ to form an 'image'**
  - 64x64 bins (~0.1 η/ϕ towers) or 224x224
- **Also try 3 'channels' (à la RGB images)[1]:**
  - Energy in Electromagnetic and Hadronic Calorimeters and no. of tracks in bin
- **Reconstruct jets using same algorithm as physics analysis (AntiKt R=1.0 trimmed) for benchmark comparison and pre-selection**





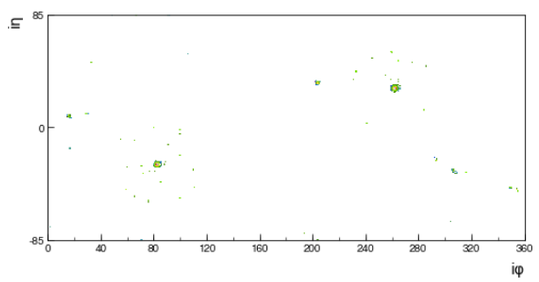
[1]Similar to Komiske, Metodiev, and Schwartz arXiv:1612.01551

U.S. DEPARTMENT OF ENERGY | Office of Science

- 4 -

Motivation

Typical Event Classification



Sensor Data

↓ Physics Reconstruction—lossy?

	Energy	p_T	R9	$\sigma_{i\eta i\eta}$	…
1	31.6	20.3	0.85	0.79	
2	45.2	37.1	0.92	0.83	
..					

Kinematic Data

↓ Classification

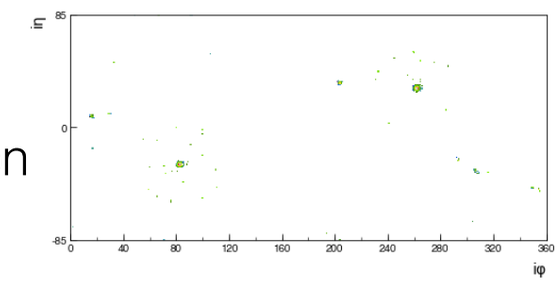
Class X
(e.g. Signal or Bkgnd)

Class Output

2

End-to-end

Event Classification



Sensor Data

↓ End-to-end Classification

Class X
(e.g. Signal or Bkgnd)

Class Output

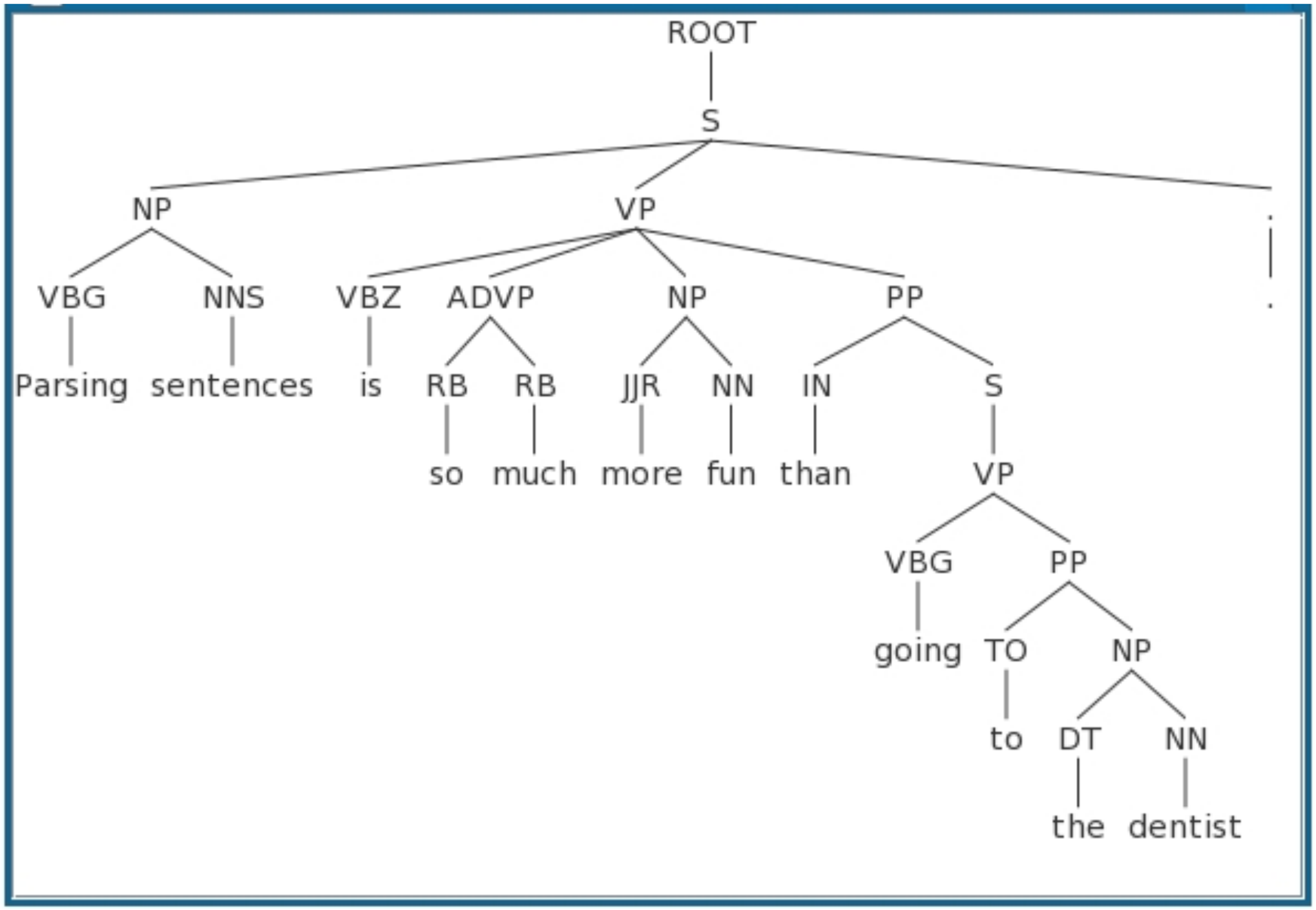
3

Physics-Aware Machine Learning

FROM IMAGES TO SENTENCES

Recursive Neural Networks showing great performance for Natural Language Processing tasks

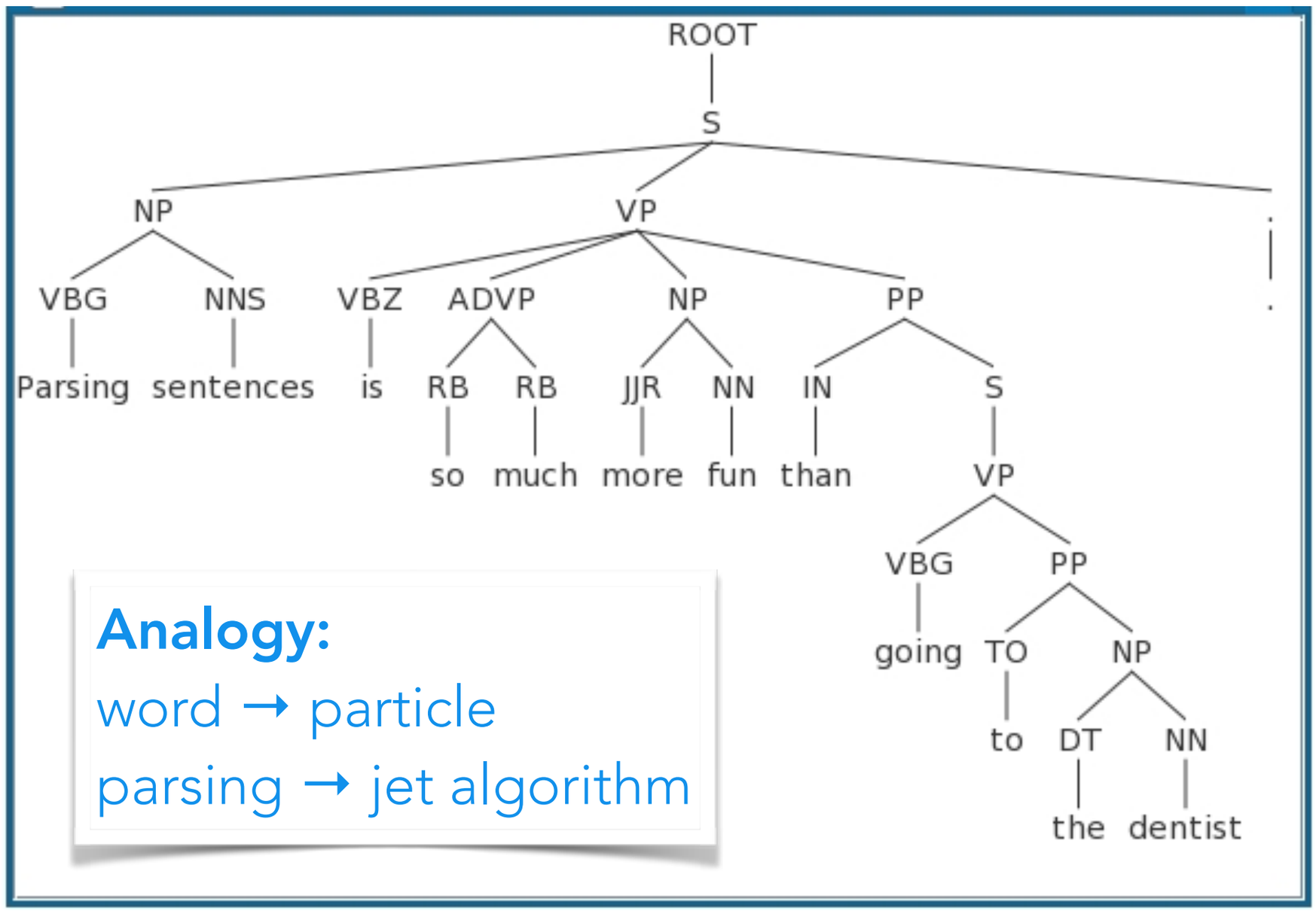
- neural network's topology given by parsing of sentence!



FROM IMAGES TO SENTENCES

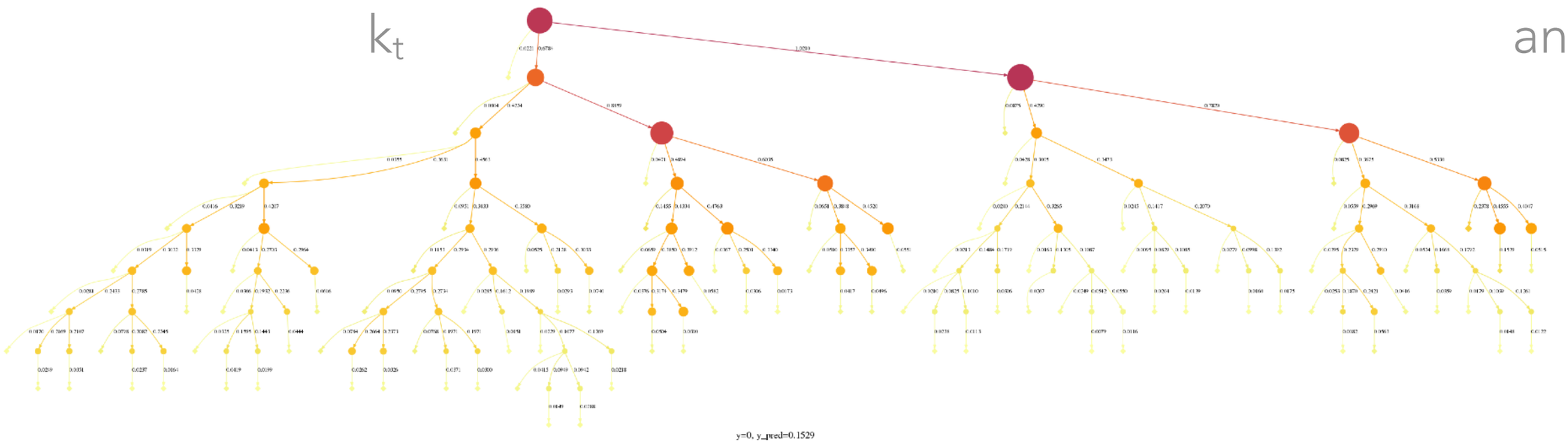
Recursive Neural Networks showing great performance for Natural Language Processing tasks

- neural network's topology given by parsing of sentence!



Analogy:
word → particle
parsing → jet algorithm

QCD-INSPIRED RECURSIVE NEURAL NETWORKS



Work with Gilles Louppe, Kyunghyun Cho, Cyril Becot

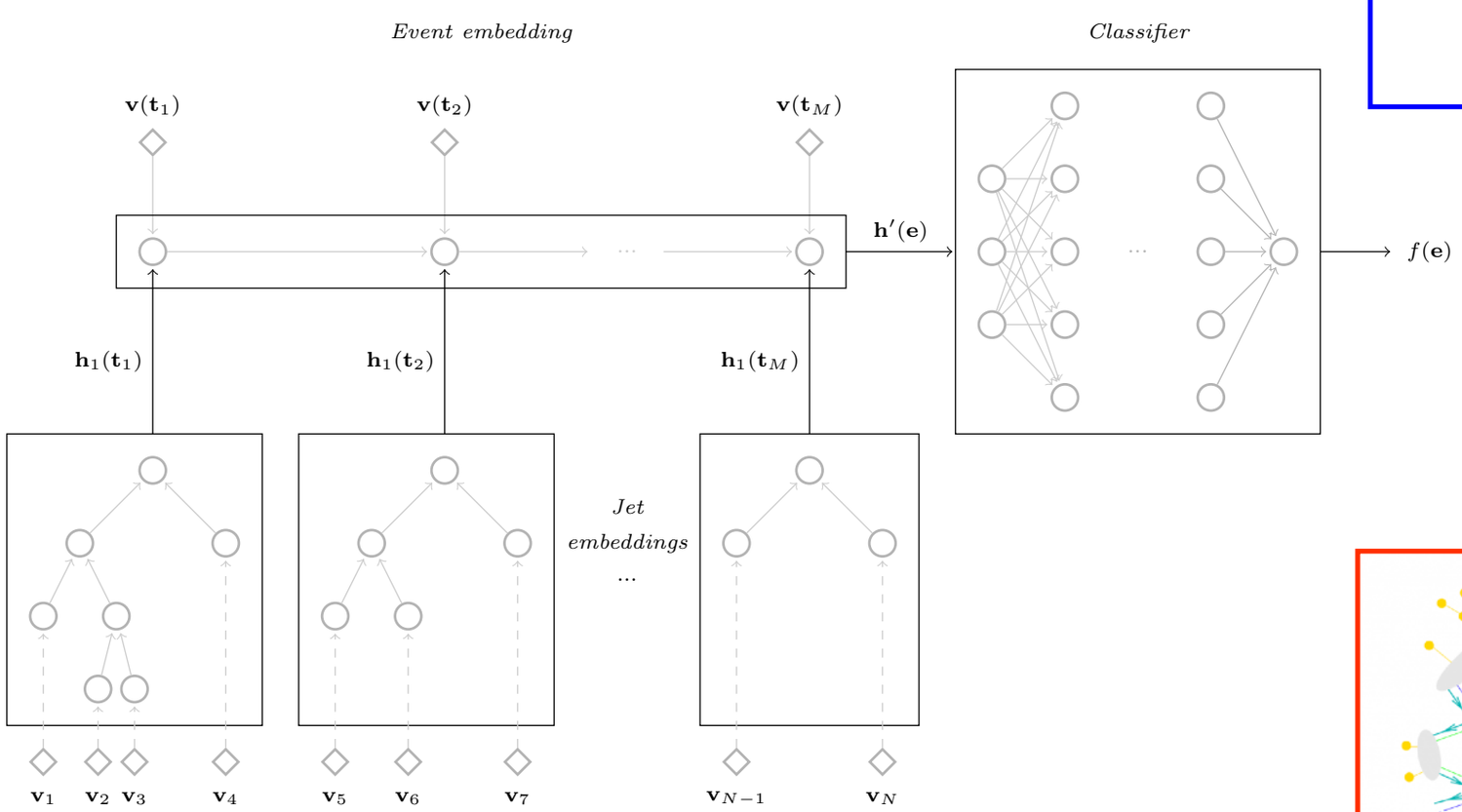
- Use sequential recombination jet algorithms to provide network topology (on a per-jet basis)
- path towards ML models with good physics properties
- Top node of recursive network provides a fixed-length **embedding** of a jet that can be fed to a classifier

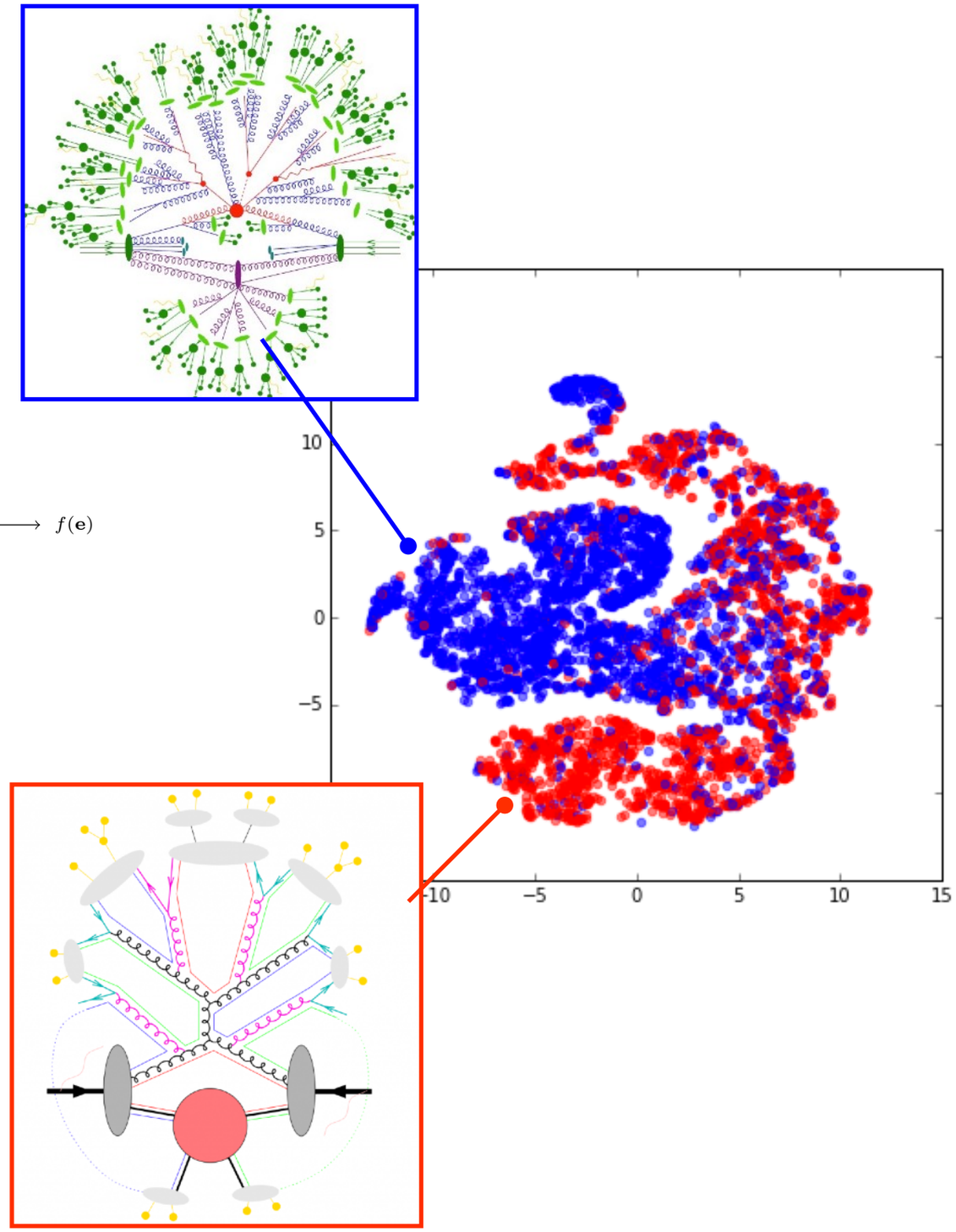
arXiv:1702.00748 & follow up work with Joan Bruna using graph conv nets

JOINTLY OPTIMIZE HIERARCHICAL MODEL

particle embedding → jet embedding → event embedding → classifier

It scales!





arXiv:1702.00748

Reconstruction as a Structured Inverse Problem

(Monte Carlo Truth = Latent Variables)

"LA MIA PARABOLA"

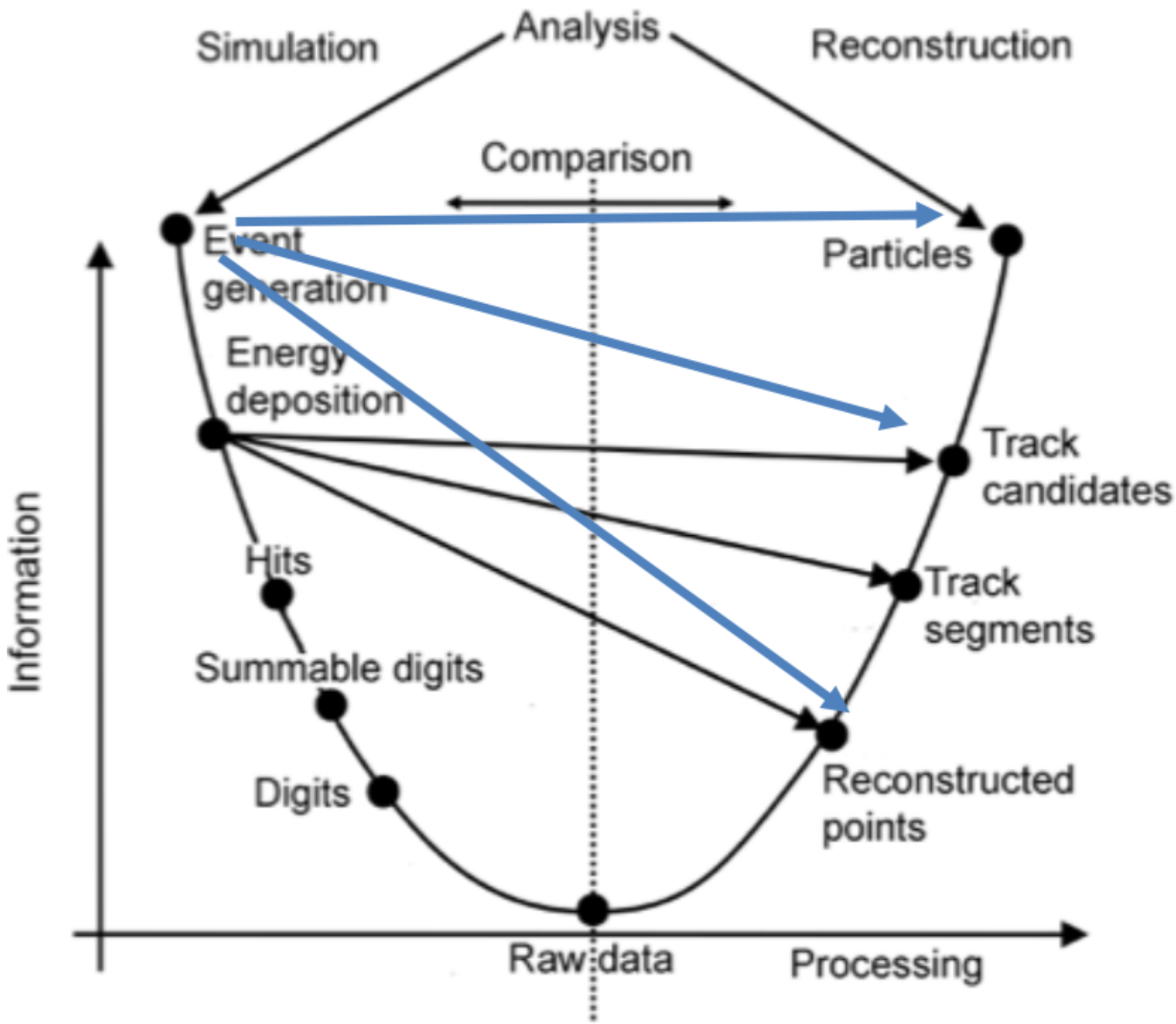
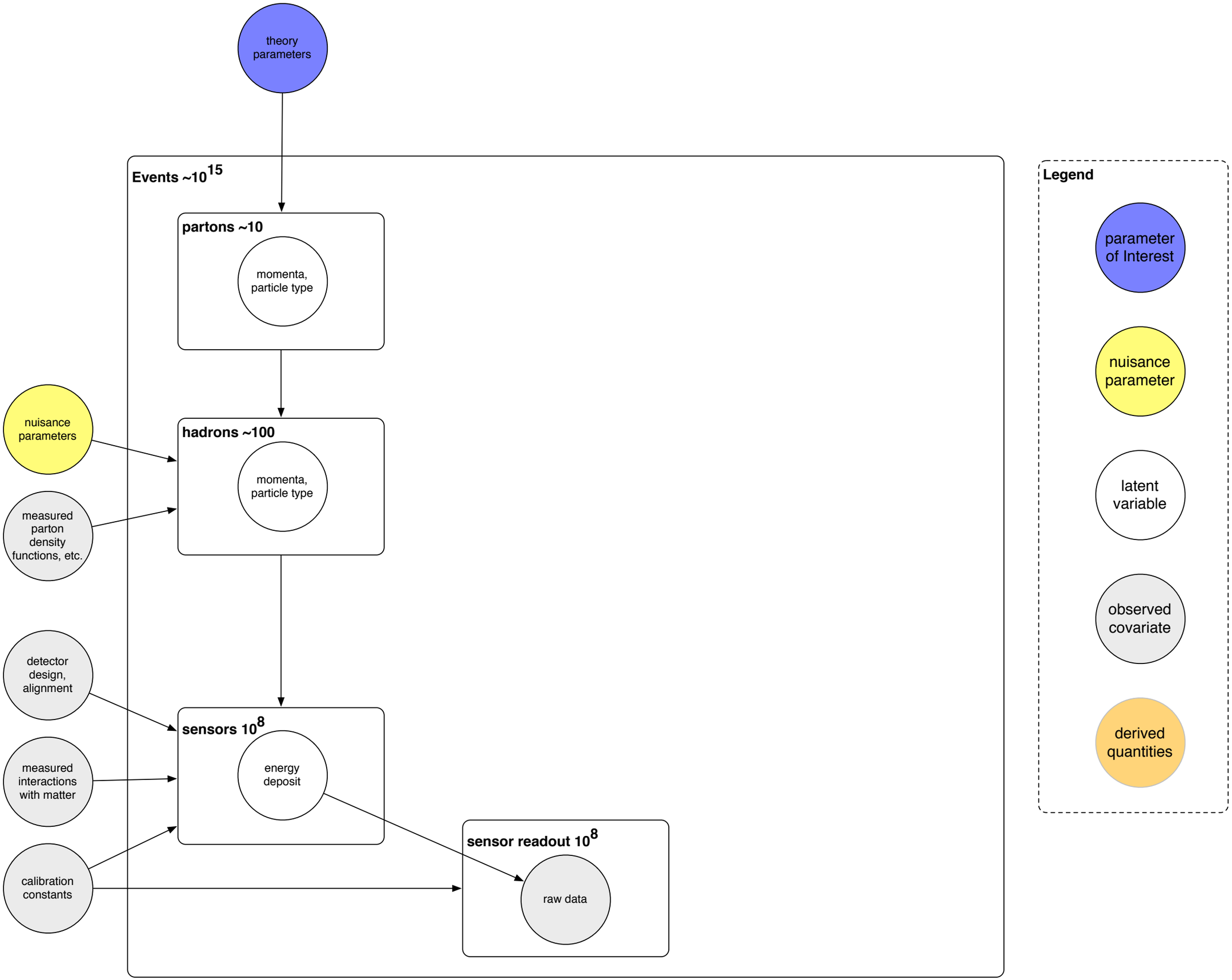
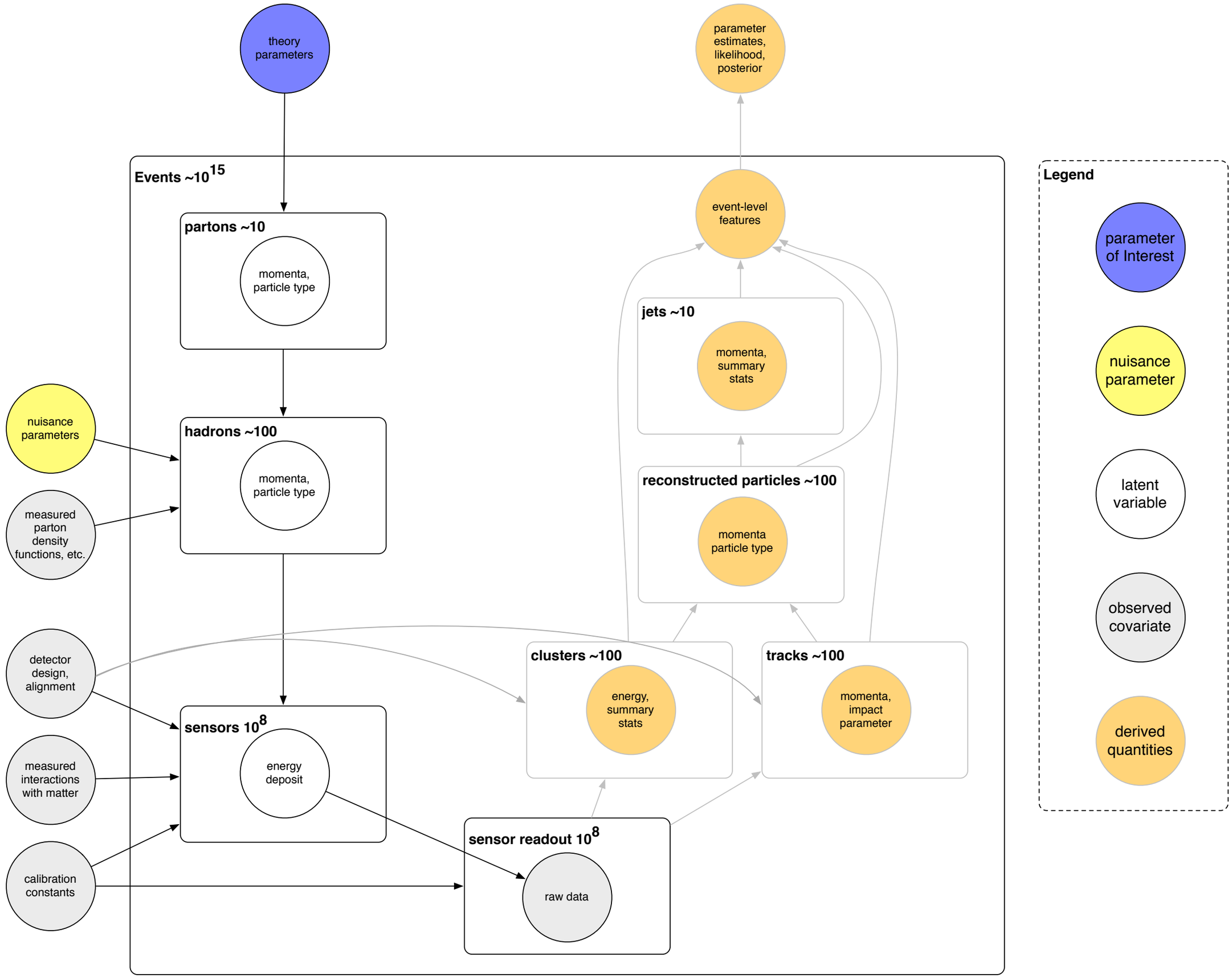


Figure by Federico Carminati, independent parallel inventions by Vincenzo Innocente & K.C.

FULL SIMULATION



FULL SIMULATION + RECONSTRUCTION



ML2.0?

How do these fit together?

Combine many of these ideas:

Large model, but **sparsely activated**

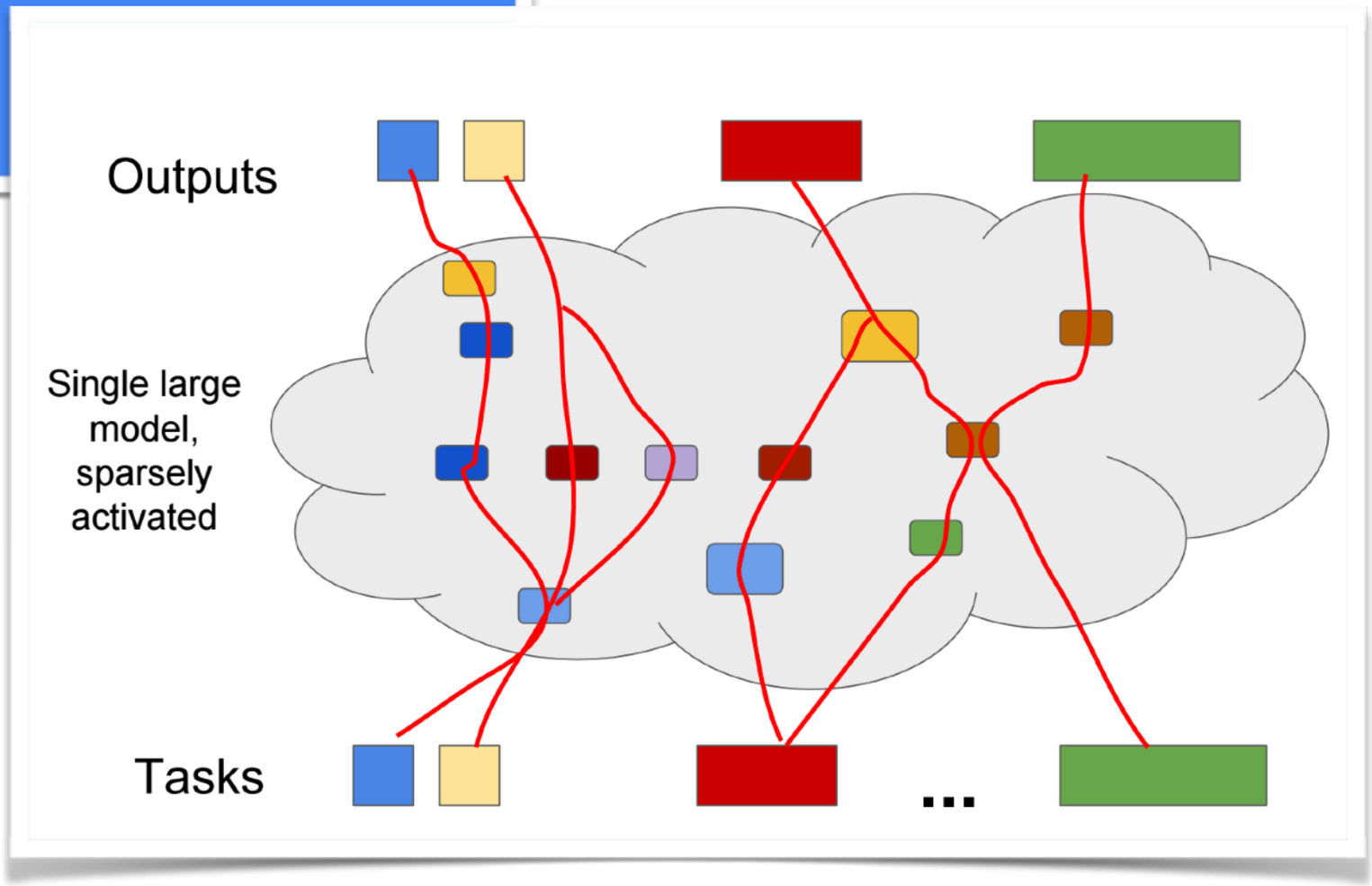
Single model to **solve many tasks** (100s to 1Ms)

Dynamically learn and **grow pathways** through large model

Hardware **specialized for ML supercomputing**

ML for efficient mapping onto this hardware

Google



Slides from Jeff Dean of Google Brain @ Jeju July 2017

https://drive.google.com/file/d/0B8z5oUpB2DysZWF1RTFuX1NEZUk/view

WHAT IS THE OBJECTIVE?

ML: What is the problem you are trying to solve?

Physicist: [eventually describes problem and formalizes objective]

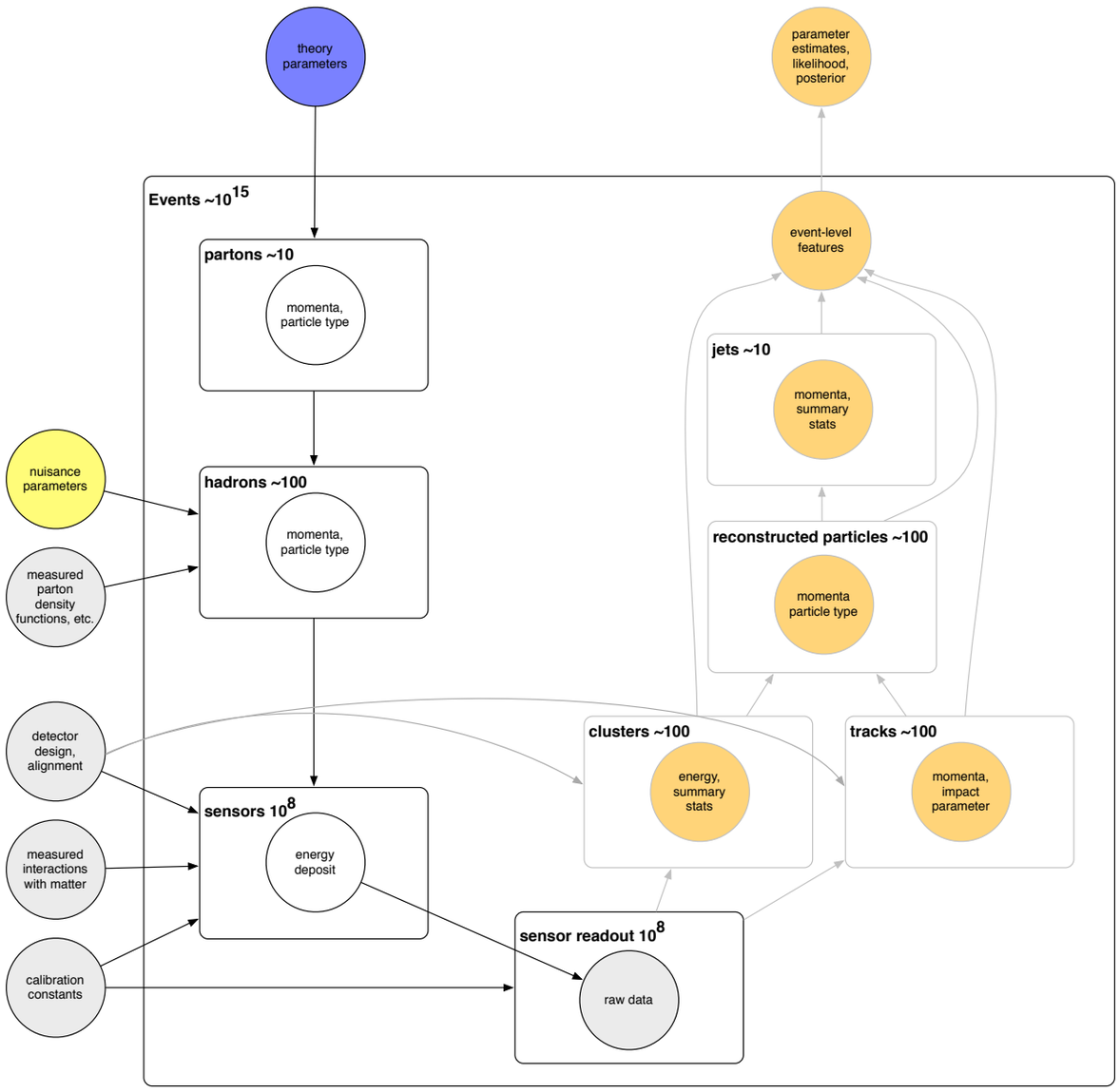
ML: Ok, well let's optimize this directly …

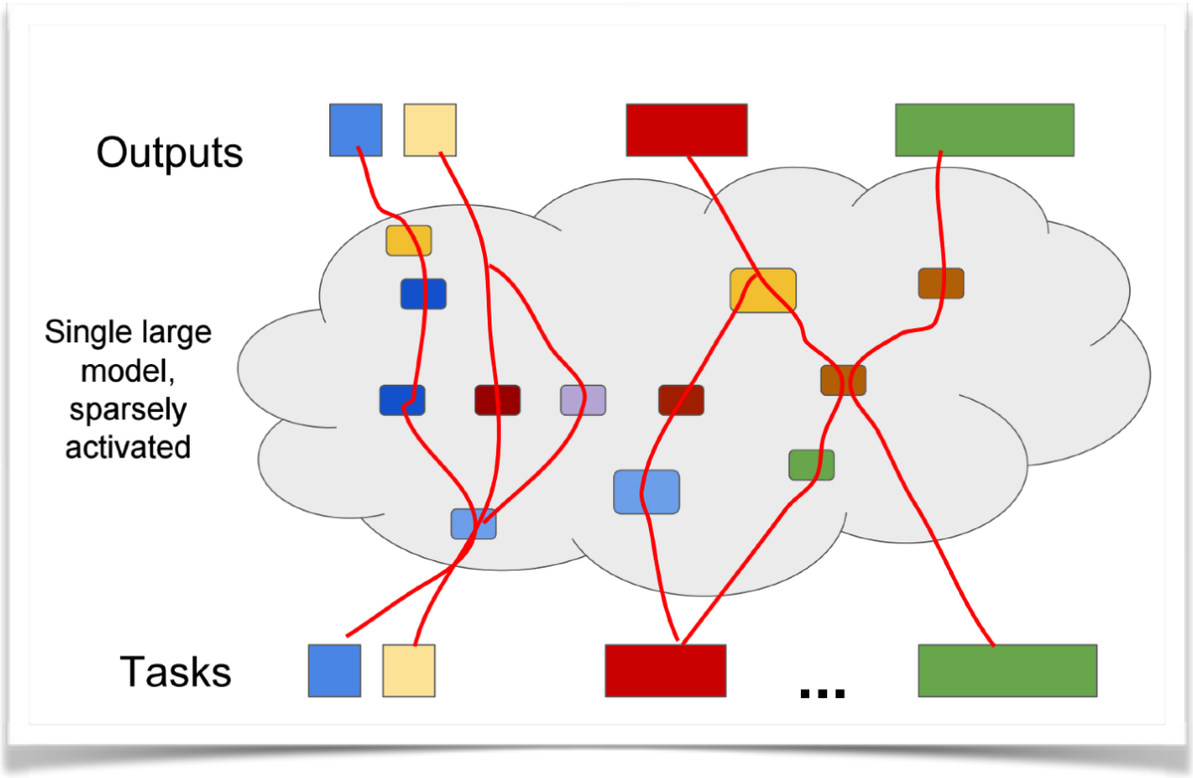
Physicist: but, I also want….

Used to criticize physicists for constantly changing problem statement, but traditional approach to physics problems has many advantages

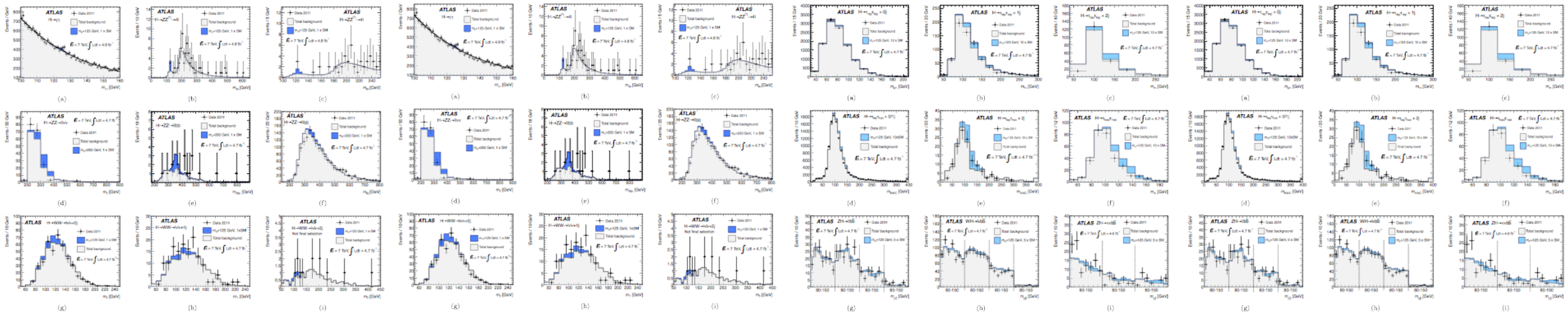
- modular, reusable components (facilitates transfer learning, "ML2.0")
- interpretable & individually validated
- a form of structural regularization

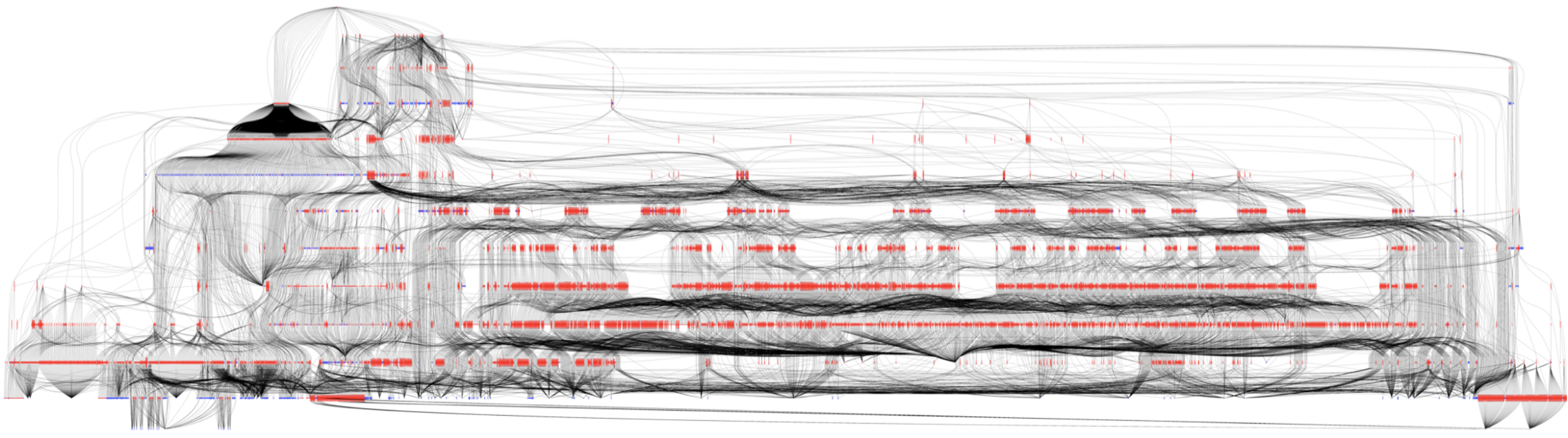
A COMMON VISION





EXTENDING OUR STATISTICAL FORMALISM WITH ML





$$\mathbf{f}_{\text{tot}}(\mathcal{D}_{\text{sim}}, \mathcal{G}|\boldsymbol{\alpha}) = \prod_{c \in \text{channels}} \left[\text{Pois}(n_c|\nu_c(\boldsymbol{\alpha})) \prod_{e=1}^{n_c} f_c(x_{ce}|\boldsymbol{\alpha}) \right] \cdot \prod_{p \in \mathbb{S}} f_p(a_p|\alpha_p)$$

THE PLAYERS

forward modeling
generation
simulation

PREDICTION

θ
parameters of interest

$p(x, z \mid \theta, \nu)$

x
observed data
simulated data

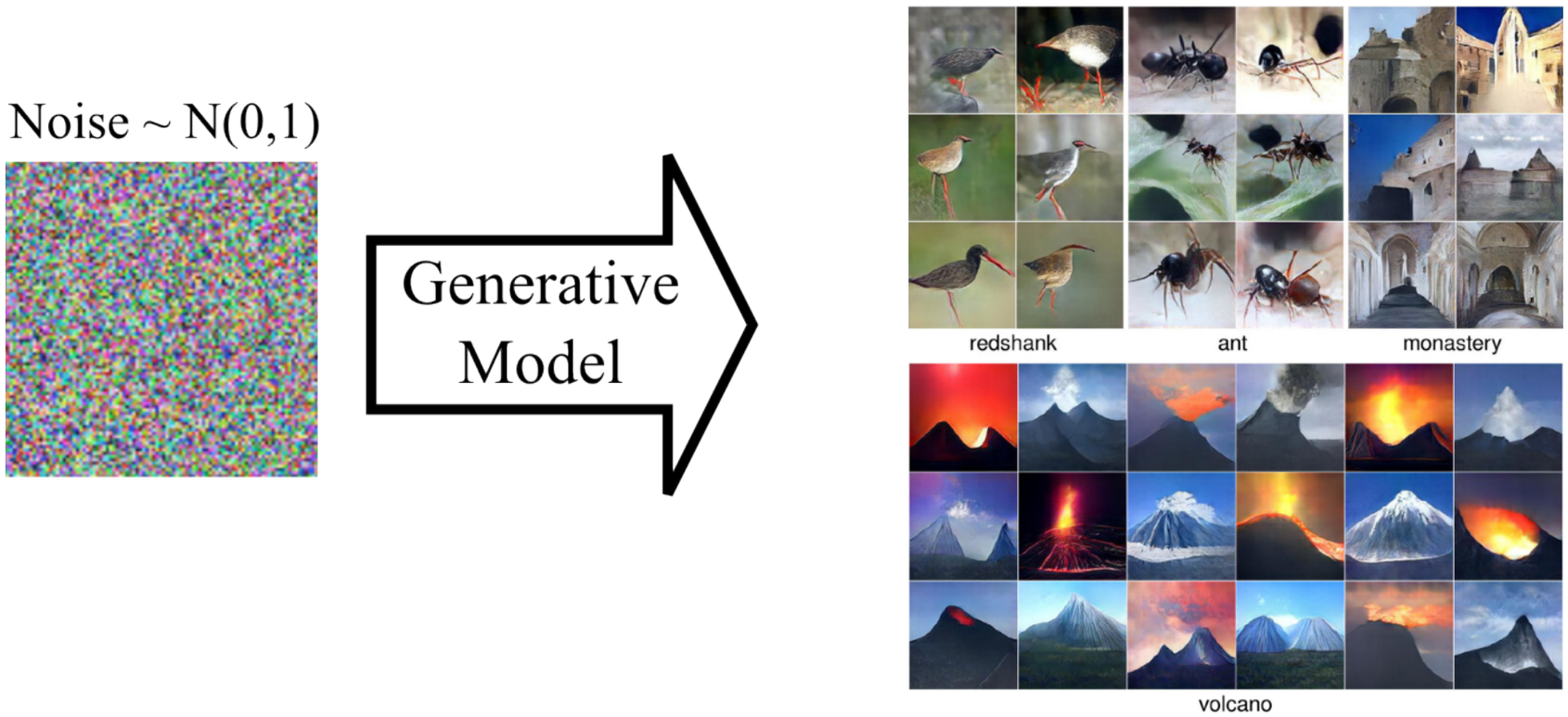
z
latent variables
Monte Carlo truth

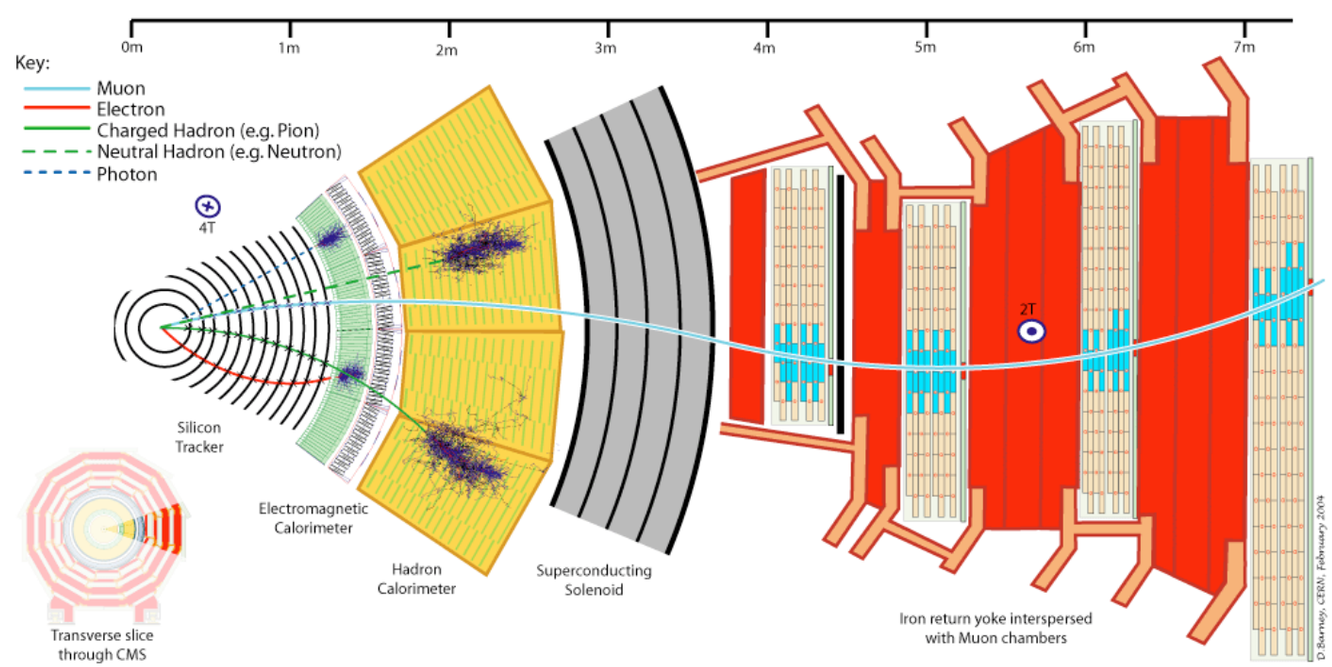
ν
nuisance parameters

INFERENCE

inverse problem
measurement
parameter estimation

LEARNING THE (SIMULATED) DATA DISTRIBUTION





http://torch.ch/blog/2015/11/13/gan.html

GENERATIVE ADVERSARIAL NETWORKS

Learning Particle Physics by Example:

Generative Adversarial Networks for Simulation

arXiv:1701.05927
arXiv:1705.02355

Luke de Oliveira
Founder, Vai Technologies
Visiting Researcher, Lawrence Berkeley National Lab

@lukede0
lukedeo@vaitech.io
https://ldo.io

with

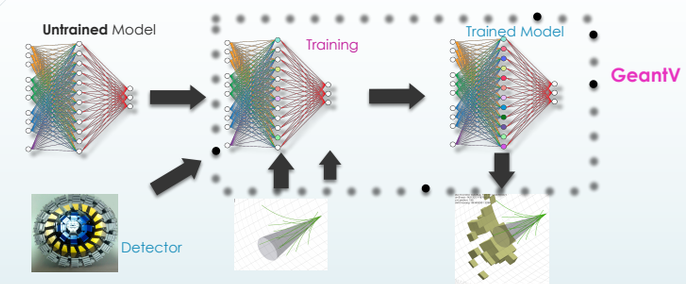
Michela Paganini and **Ben Nachman**
Yale, LBNL LBNL

Generative models for fast simulation

Sofia Vallecorsa
for the GeantV project

28

DL engine for fast simulation in GeantV



3d GAN represent first proof of concept

- We aim at a generic fully configurable tool
- Optimal network design depends on the problem to solve
  - Embedded algorithms for hyper-parameters tuning and meta-optimization
  - Parallelization on CPU/GPU clusters

SCHOOL OF DATA ANALYSIS LAMBDA Yandex

Modeling detector digitization and read-out with adversarial networks

ACAT, Seattle, 2017-08-21

Maxim Borisyak[1], Chase Shimmin[3], Andy Nelson[2], Andrey Ustyuzhanin[1]

[1] Yandex, NRU Higher School of Economics
[2] University of California Irvine
[3] Yale University

GENERATIVE ADVERSARIAL NETWORKS

Learning Particle Physics by Example:

Generative Adversarial Networks for Simulation

arXiv:1701.05927
arXiv:1705.02355

Luke de Oliveira
Founder, Vai Technologies
Visiting Researcher, Lawrence Berkeley National Lab

@lukede0
lukedeo@vaitech.io
https://ldo.io

with

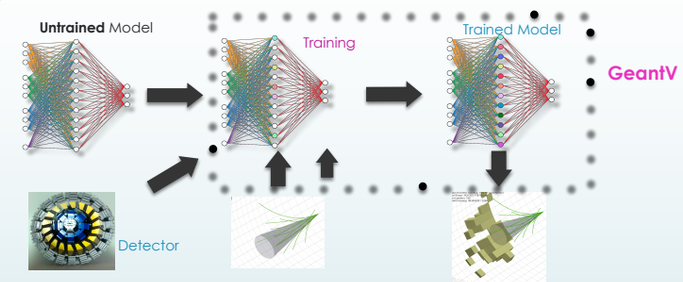
Michela Paganini and **Ben Nachman**
Yale, LBNL LBNL

Generative models for fast simulation

Sofia Vallecorsa
for the GeantV project

28

DL engine for fast simulation in GeantV



3d GAN represent first proof of concept

- We aim at a generic fully configurable tool
- Optimal network design depends on the problem to solve
  - Embedded algorithms for hyper-parameters tuning and meta-optimization
  - Parallelization on CPU/GPU clusters

SCHOOL OF DATA ANALYSIS LAMBDA Yandex

Modeling detector digitization and read-out with adversarial networks

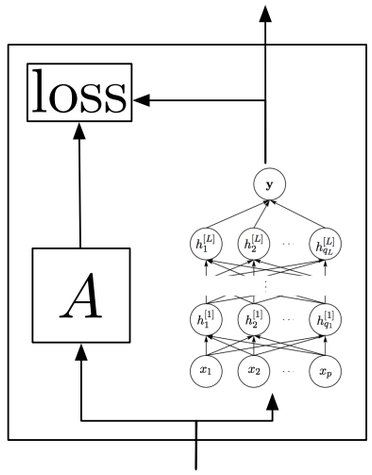
ACAT, Seattle, 2017-08-21

Maxim Borisyak[1], Chase Shimmin[3], Andy Nelson[2], Andrey Ustyuzhanin[1]

[1] Yandex, NRU Higher School of Economics
[2] University of California Irvine
[3] Yale University

NN Approximation of a primitive function

- Wrap a (un)known function
- The wrapper contains both the known function and a NN that approximates it
- Intermediate loss is defined as the discrepancy between the known function and the NN
- Output may be either the known function's and NN's output



Kyunghyun Cho

THE PLAYERS

forward modeling
generation
simulation

PREDICTION

θ
parameters of interest

$p(x, z \mid \theta, \nu)$

x
observed data
simulated data

z
latent variables
Monte Carlo truth

ν
nuisance parameters

INFERENCE

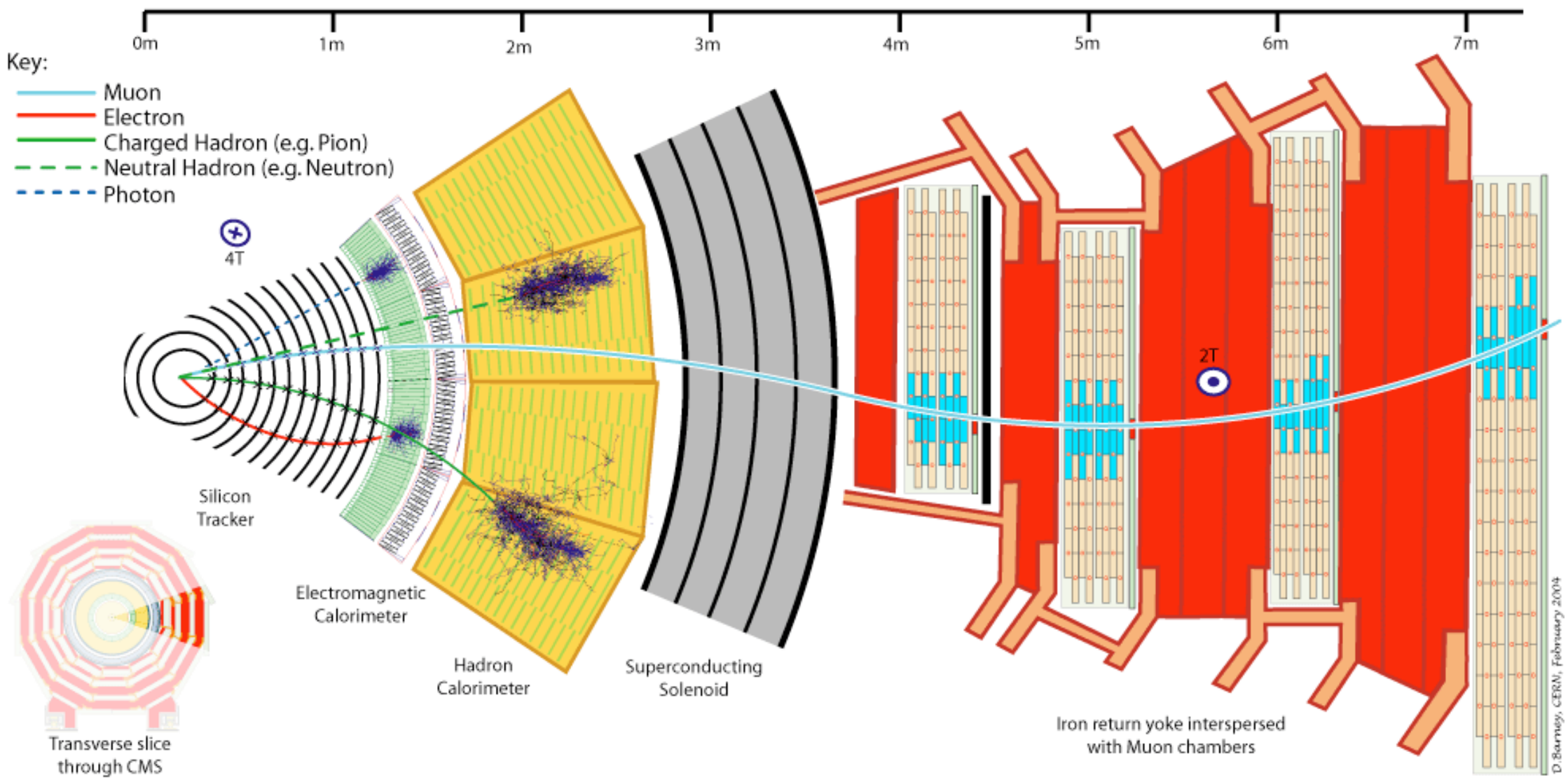
inverse problem
measurement
parameter estimation

DETECTOR SIMULATION

Conceptually: Prob(detector response | particles)

Implementation: Monte Carlo integration over micro-physics

Consequence: evaluation of the likelihood is intractable



DETECTOR SIMULATION

Conceptually: Prob(detector response | particles)

Implementation: Monte Carlo integration over micro-physics

Consequence: evaluation of the likelihood is intractable

This motivates a new class of algorithms for what is called **likelihood-free inference**, which only require ability to generate samples from the simulation in the "forward mode"

A COMMON THEME

ABC

resources on approximate Bayesian computational methods

Search

Home

Home

This website keeps track of developments in approximate Bayesian computation (ABC) (a.k.a. likelihood-free), a class of computational statistical methods for Bayesian inference under intractable likelihoods. The site is meant to be a resource both for biologists and statisticians who want to learn more about ABC and related methods. Recent publications are under Publications 2012. A comprehensive list of publications can be found under Literature. If you are unfamiliar with ABC methods see the Introduction. Navigate using the menu to learn more.

ABC in Montreal | ABC in Montreal (2014)

ABC in Montreal

Approximate Bayesian computation (ABC) or likelihood-free (LF) methods have developed mostly beyond the radar of the machine learning community, but are important tools for a large and diverse segment of the scientific community. This is particularly true for systems and population biology, computational neuroscience, computer vision, healthcare sciences, but also many others.

Interaction between the ABC and machine learning community has recently started and contributed to important advances. In general, however, there is still significant room for more intense interaction and collaboration. Our workshop aims at being a place for this to happen.

ICML 2017 Workshop on Implicit Models

Workshop Aims

Probabilistic models are an important tool in machine learning. They form the basis for models that generate realistic data, uncover hidden structure, and make predictions. Traditionally, probabilistic models in machine learning have focused on prescribed models. Prescribed models specify a joint density over observed and hidden variables that can be easily evaluated. The requirement of a tractable density simplifies their learning but limits their flexibility --- several real world phenomena are better described by simulators that do not admit a tractable density. Probabilistic models defined only via the simulations they produce are called implicit models.

Arguably starting with generative adversarial networks, research on implicit models in machine learning has exploded in recent years. This workshop's aim is to foster a discussion around the recent developments and future directions of implicit models.

Implicit models have many applications. They are used in ecology where models simulate animal populations over time; they are used in phylogeny, where simulations produce hypothetical ancestry trees; they are used in physics to generate particle simulations for high energy processes. Recently, implicit models have been used to improve the state-of-the-art in image and content generation. Part of the workshop's focus is to discuss the commonalities among applications of implicit models.

Of particular interest at this workshop is to unite fields that work on implicit models. For example:

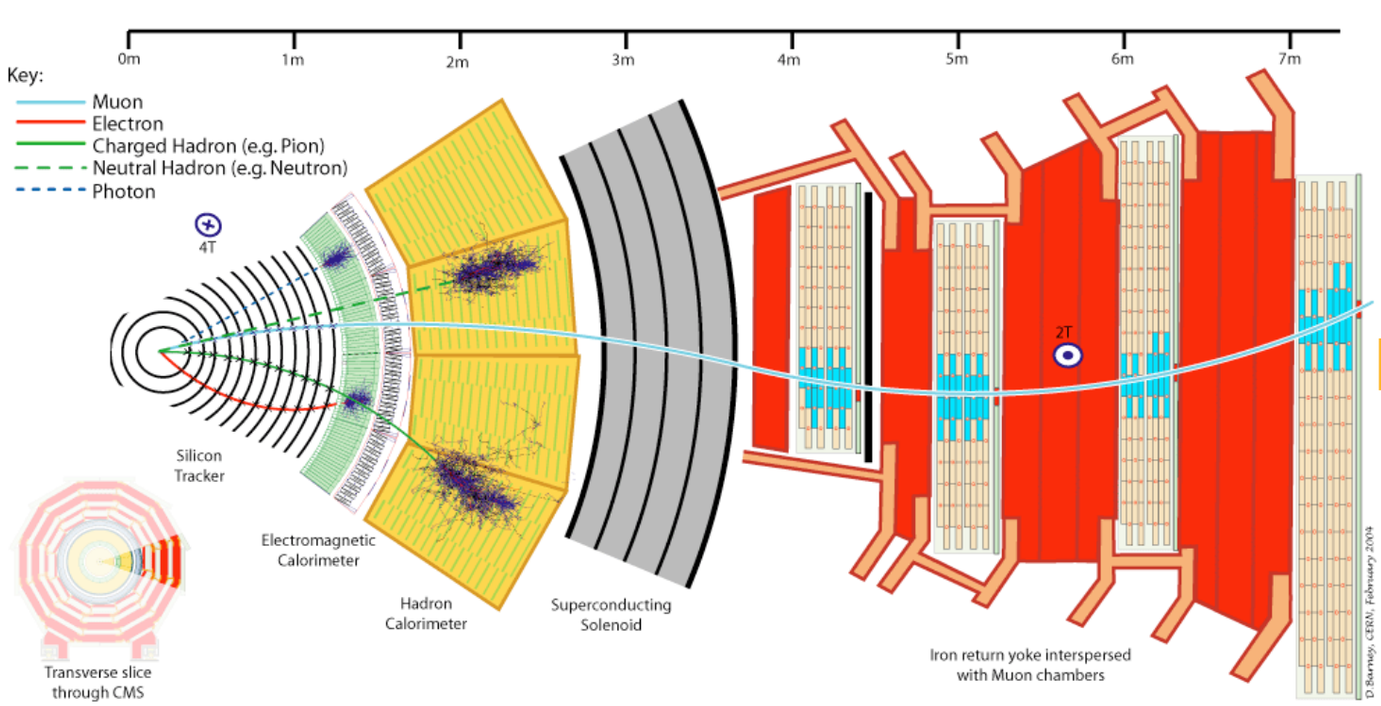
- **Generative adversarial networks** (a NIPS 2016 workshop) are implicit models with an adversarial training scheme.
- Recent advances in **variational inference** (a NIPS 2015 and 2016 workshop) have leveraged implicit models for more accurate approximations.
- **Approximate Bayesian computation** (a NIPS 2015 workshop) focuses on posterior inference for models with implicit likelihoods.
- Learning implicit models is deeply connected to **two sample testing, density ratio and density difference** estimation.

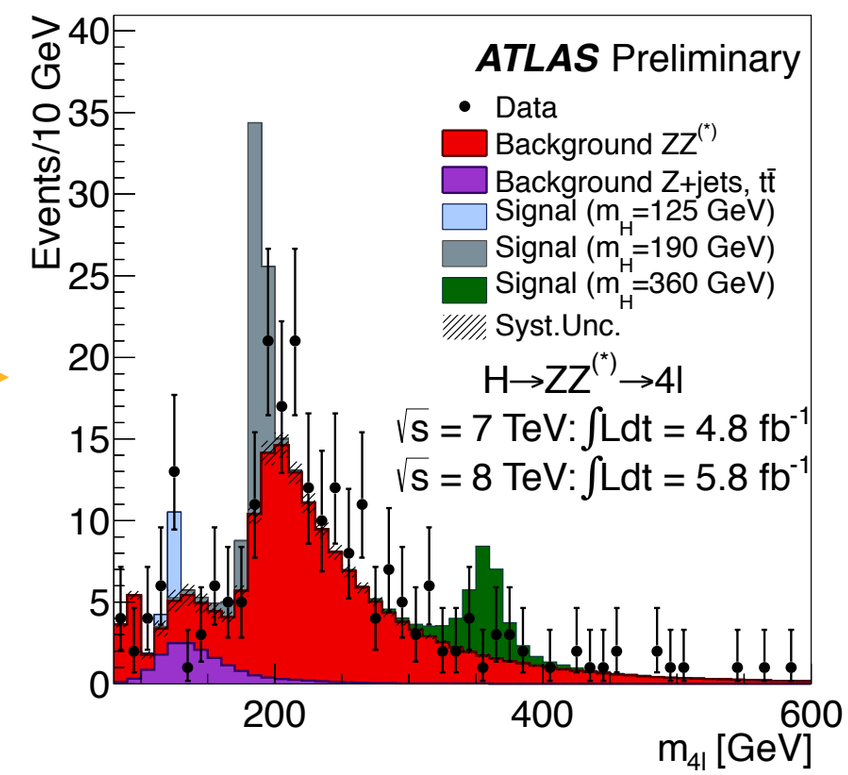
We hope to bring together these different views on implicit models, identifying their core challenges and combining their innovations.

10^8 SENSORS → 1 REAL-VALUED QUANTITY

Most measurements and searches for new particles at the LHC are based on the distribution of a single variable or feature

- choosing a good variable (feature engineering) is a task for a skilled physicist and tailored to the goal of measurement or new particle search
- likelihood $p(x|\theta)$ **approximated** using histograms (univariate density estimation)

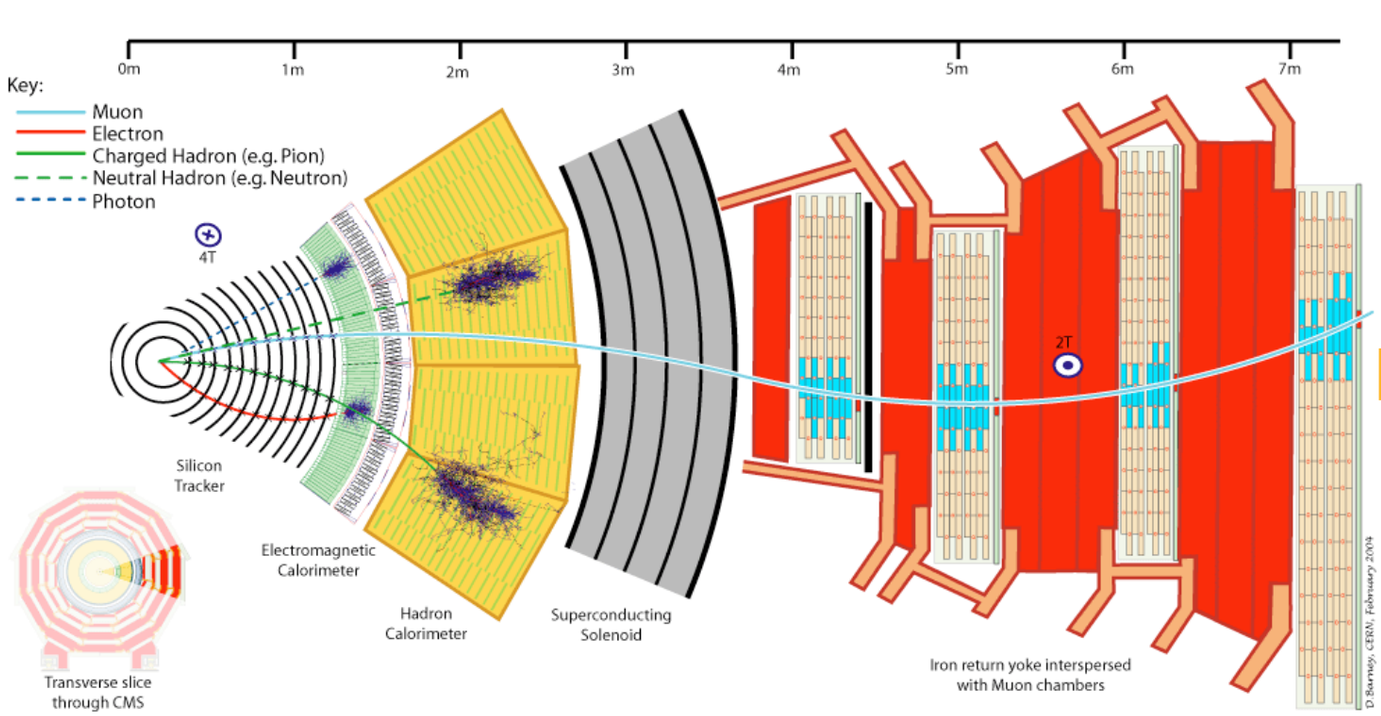


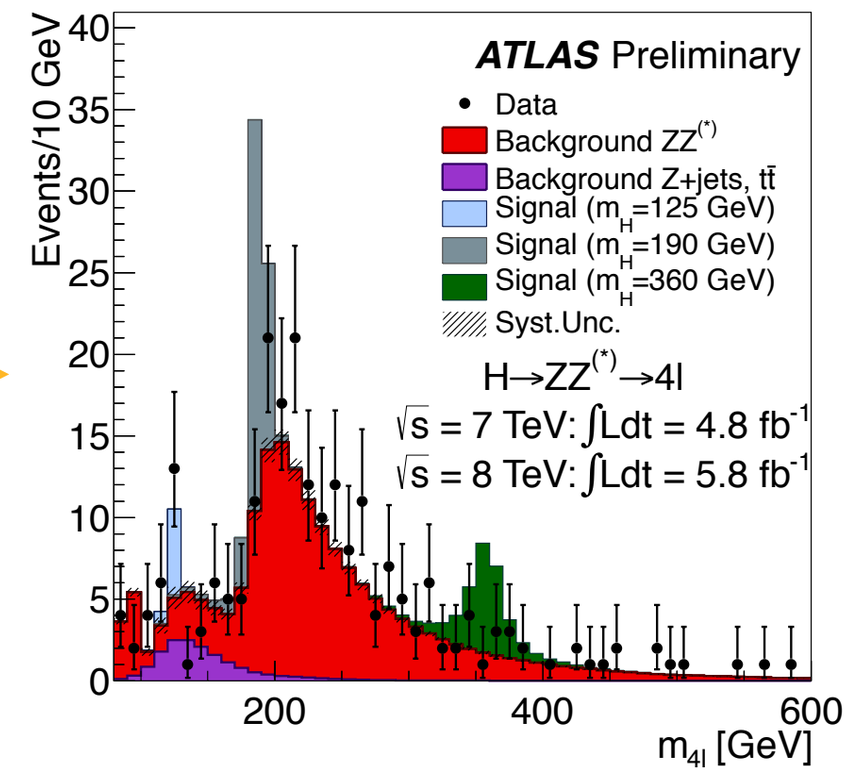


10^8 SENSORS → 1 REAL-VALUED QUANTITY

Most measurements and searches for new particles at the LHC are based on the distribution of a single variable or feature

- choosing a good variable (feature engineering) is a task for a skilled physicist and tailored to the goal of measurement or new particle search
- likelihood $p(x|\theta)$ **approximated** using histograms (univariate density estimation)





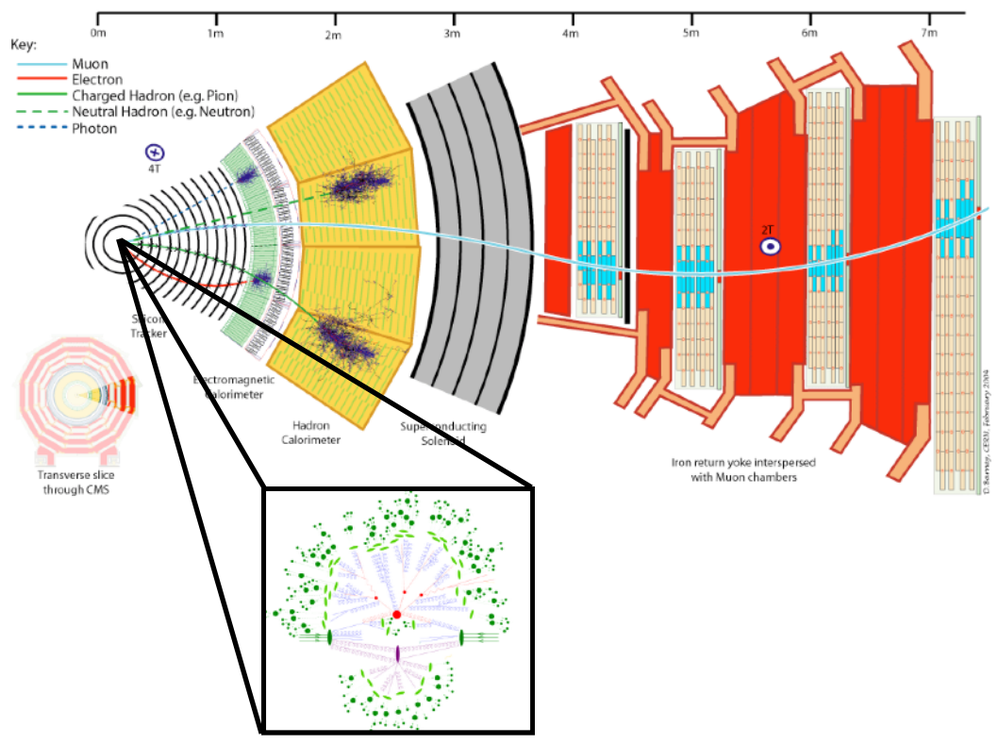
This doesn't scale if x is high dimensional!

MODEL SPECIFICATION

APPROACH	TRACTABLE LIKELIHOOD	SCALABLE TO HIGH DIM. X	COMMENT
SIMULATOR	X	√	MOST DETAILED
ANALYTIC FUNCTION	√	X	AD HOC
HISTOGRAM	√	X	TRADITIONAL SURROGATE
NEURAL NETWORK	√	√	PROMISING
GAUSSIAN PROCESS	√	√ (MODERATE)	INCLUDES UNCERTAINTY

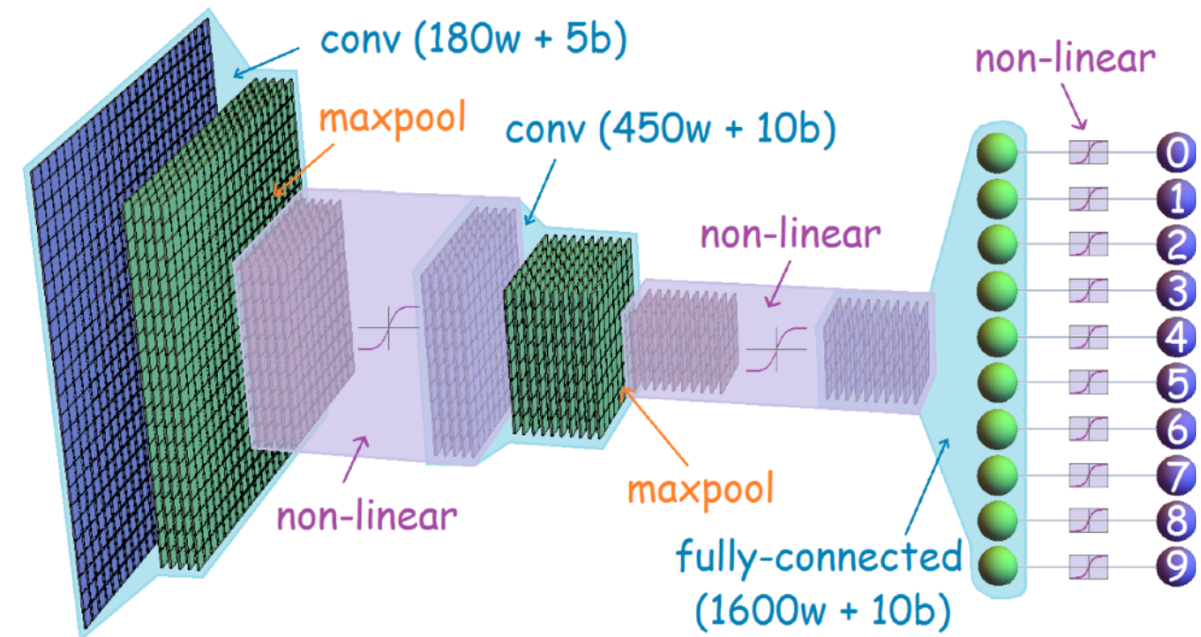
TWO APPROACHES

Use simulator
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

Learn simulator
(with deep learning)

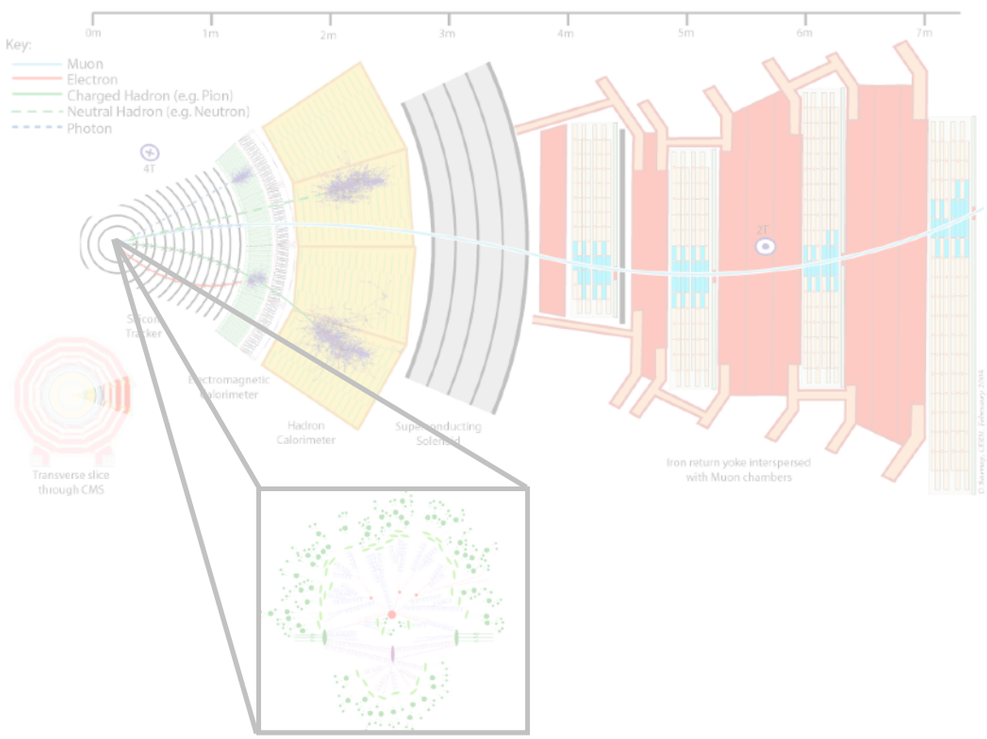


- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

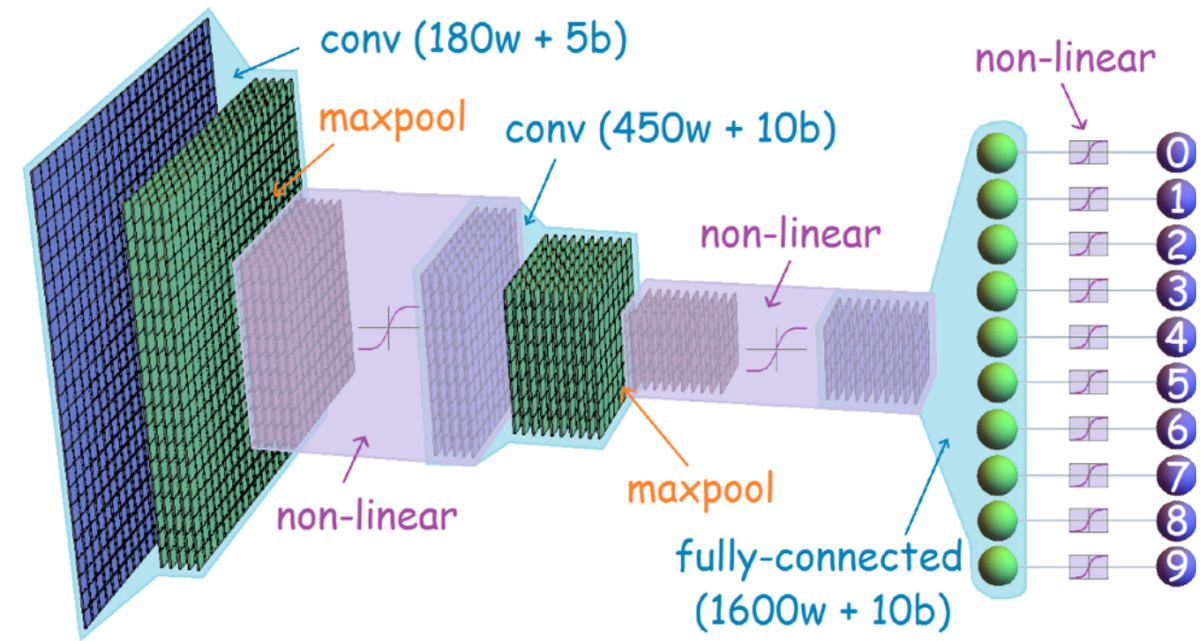
TWO APPROACHES

Use simulator
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

Learn simulator
(with deep learning)



- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

Likelihood-free inference with neural networks

Example: Higgs Effective Field Theory

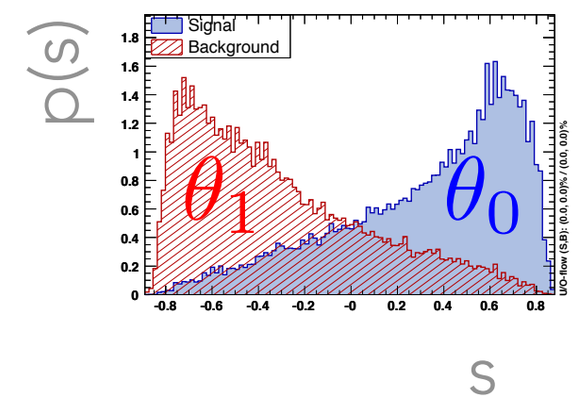
CARL

The intractable likelihood ratio based on high-dimensional features x is:

$$\frac{p(x|\theta_0)}{p(x|\theta_1)}$$

We can show that an **equivalent test** can be made from 1-D projection

$$\frac{p(x|\theta_0)}{p(x|\theta_1)} = \frac{p(s(x;\theta_0,\theta_1)|\theta_0)}{p(s(x;\theta_0,\theta_1)|\theta_1)}$$



if the scalar map s: X → $\mathbb{R}$ has the same level sets as the likelihood ratio

$$s(x;\theta_0;\theta_1) = \text{monotonic}[\ p(x|\theta_0)/p(x|\theta_1)\ ]$$

Estimating the density of $s(x;\theta_0,\theta_1)$ via the simulator calibrates the ratio.

K.C., G. Louppe, J. Pavez: http://arxiv.org/abs/1506.02169

CARL SOFTWARE

Gilles Louppe

http://diana-hep.org/carl/

diana-hep.org

DiscoveryLinks Higgs RooStats ALEPH Apple News Life Stuff ATLAS Wikipedia, inSpire Theory&Practice nyu espace JCSS HCG

Meet F Jupyter Note... Weekend rea... early-career-... 2016 Electio... 12-day Event... Joint meetin... carl API

Fork me on GitHub

Index

Sub-modules

- carl.data
- carl.distributions
- carl.learning
- carl.ratios

Notebooks

- Composing and fitting distributions
- Diagnostics for approximate likelihood ratios
- Likelihood ratios of mixtures of normals
- Parameterized inference from multidimensional data
- Parameterized inference with nuisance parameters

Display a menu

carl module

`carl` is a toolbox for likelihood-free inference in Python.

The likelihood function is the central object that summarizes the information from an experiment needed for inference of model parameters. It is key to many areas of science that report the results of classical hypothesis tests or confidence intervals using the (generalized or profile) likelihood ratio as a test statistic. At the same time, with the advance of computing technology, it has become increasingly common that a simulator (or generative model) is used to describe complex processes that tie parameters of an underlying theory and measurement apparatus to high-dimensional observations. However, directly evaluating the likelihood function in these cases is often impossible or is computationally impractical.

In this context, the goal of this package is to provide tools for the likelihood-free setup, including likelihood (or density) ratio estimation algorithms, along with helpers to carry out inference on top of these.

This project is still in its early stage of development. Join us on GitHub if you feel like contributing!

build passing | coverage 91% | DOI 10.5281/zenodo.47798

Likelihood-free inference with calibrated classifiers

Extensive details regarding likelihood-free inference with calibrated classifiers can be found in the companion paper *"Approximating Likelihood Ratios with Calibrated Discriminative Classifiers", Kyle Cranmer, Juan Pavez, Gilles Louppe.* http://arxiv.org/abs/1506.02169

Installation

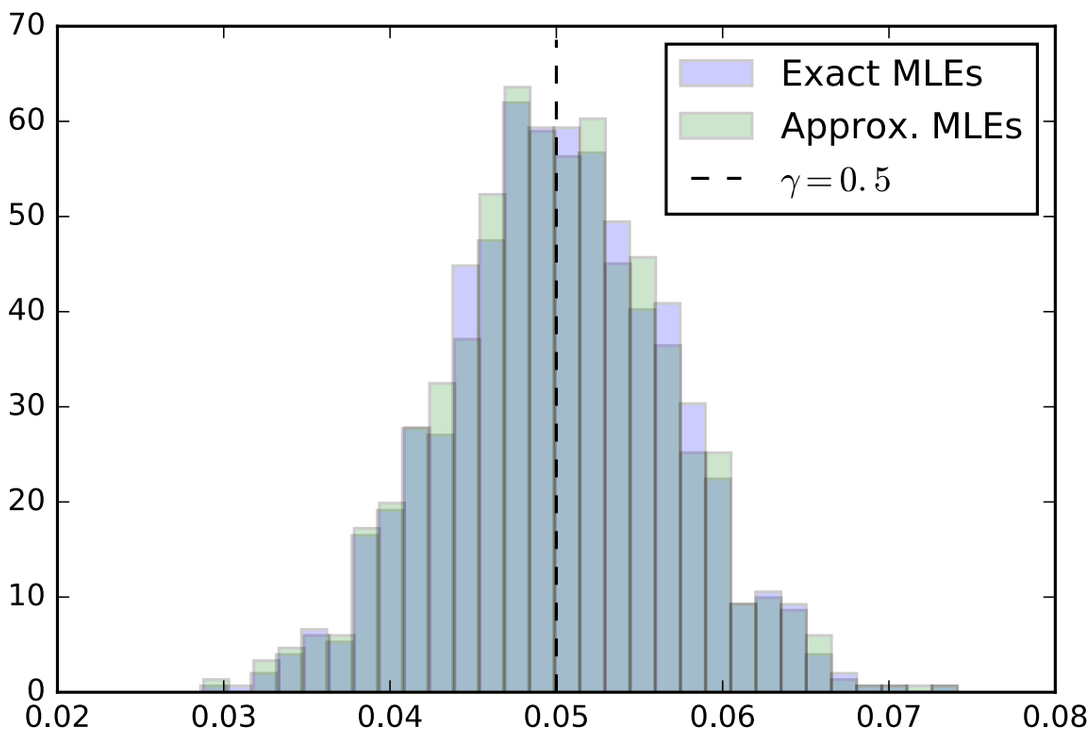
The following dependencies are required:

- Numpy >= 1.11

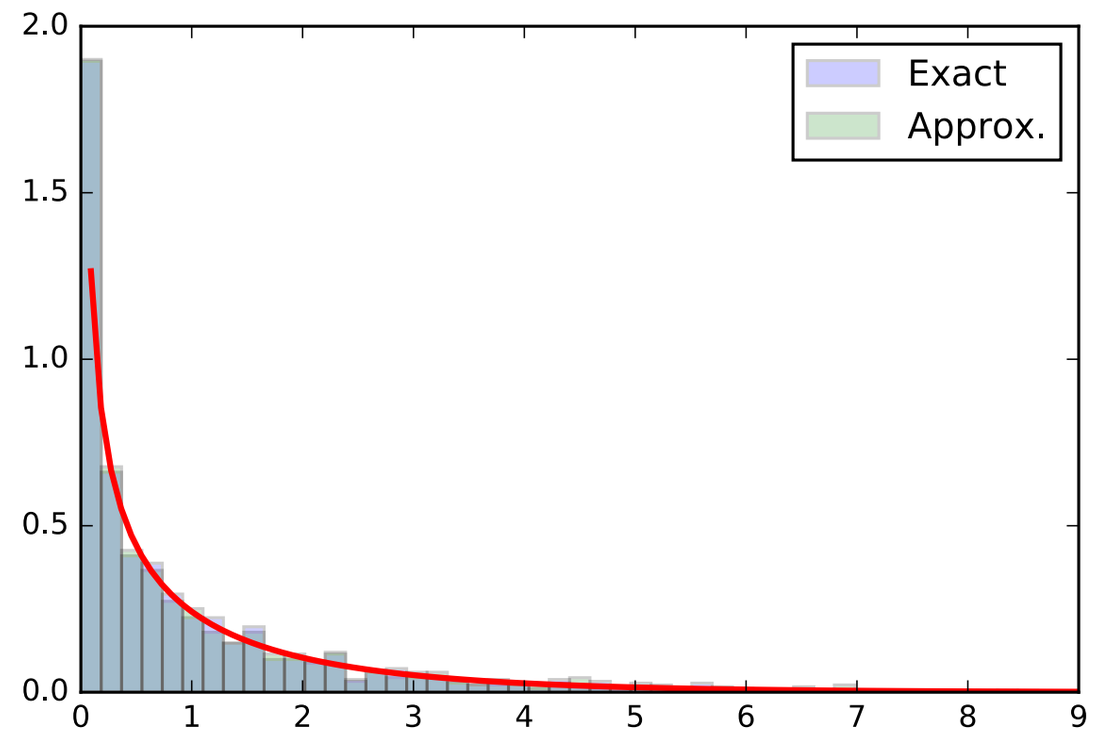
AMORTIZED LIKELIHOOD-FREE INFERENCE

Once we've learned the function s(x; θ) to approximate the likelihood, we can apply it to any data x.

- Here we check asymptotic distribution of profile likelihood ratio (Wilks's theorem)



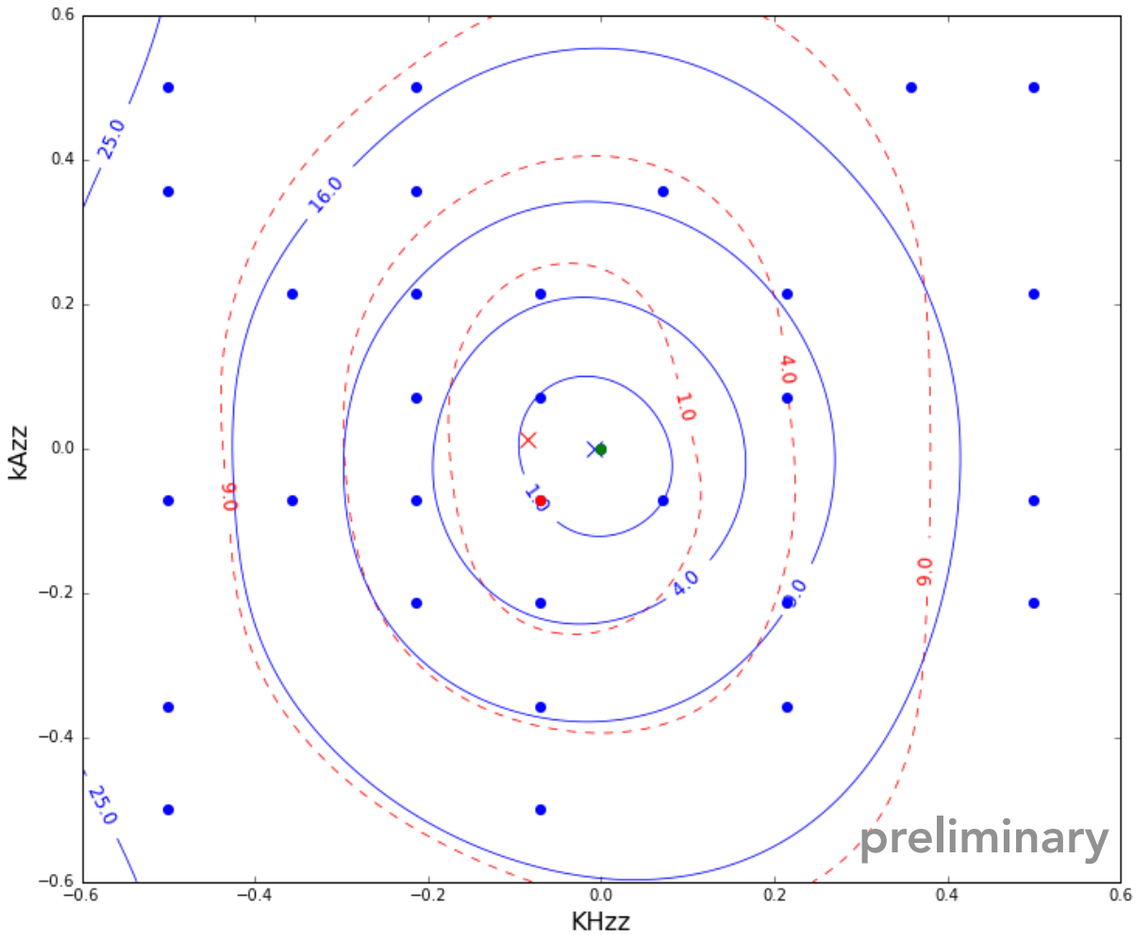
(a) Exact vs. approximated MLEs.

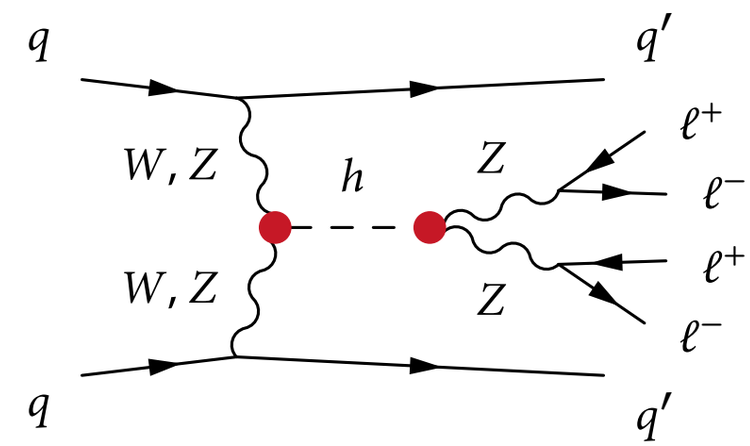


(b) $p(-2\log\Lambda(\gamma = 0.05)\,|\,\gamma = 0.05)$

APPLICATION TO THE HIGGS

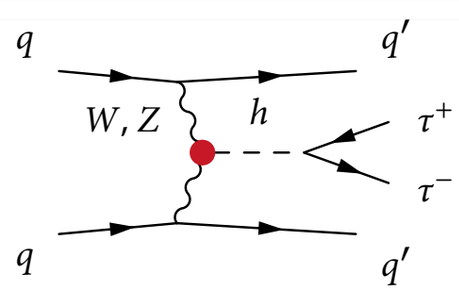
Preliminary work using fast detector simulation and CARL to approximate likelihoods using full kinematic information parametrized in 5-d coefficients of a Quantum Field Theory





16 observables
(using the CARL)

2 observables
(histogram templates)

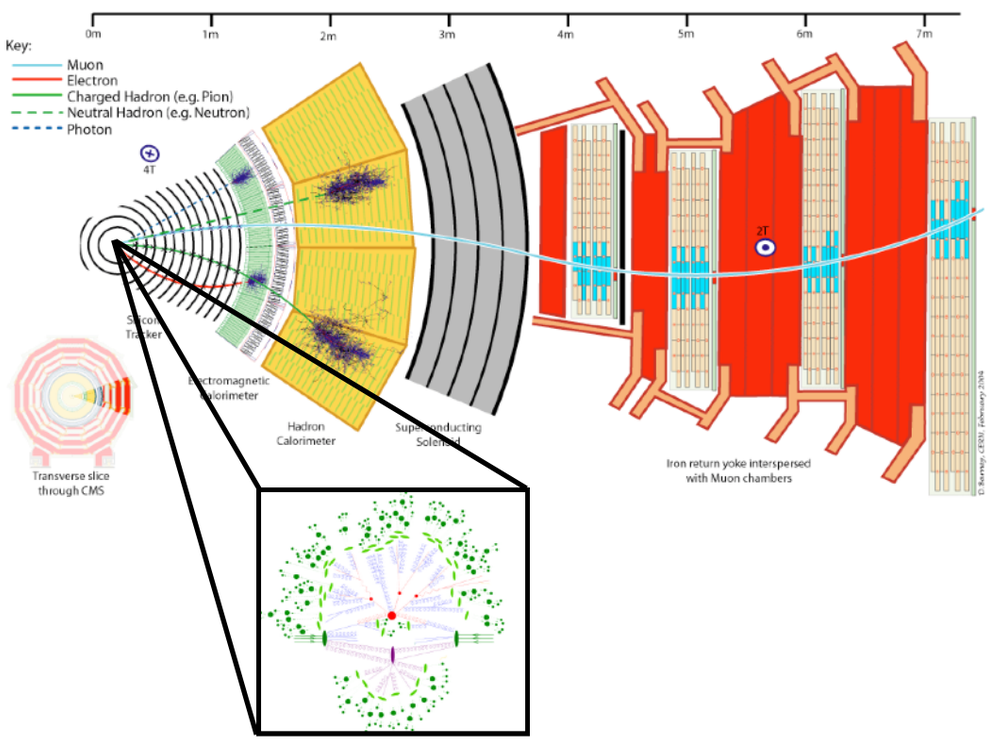


Equivalent to 3x more data.
(idealized, no systematic uncertainty)

work with Juan Pavez, Gilles Louppe, Cyril Becot, and Lukas Heinrich; Johann Brehmer, Felix Kling, and Tilman Plehn
"Better Higgs Measurements Through Information Geometry" [arXiv:1612.05261]

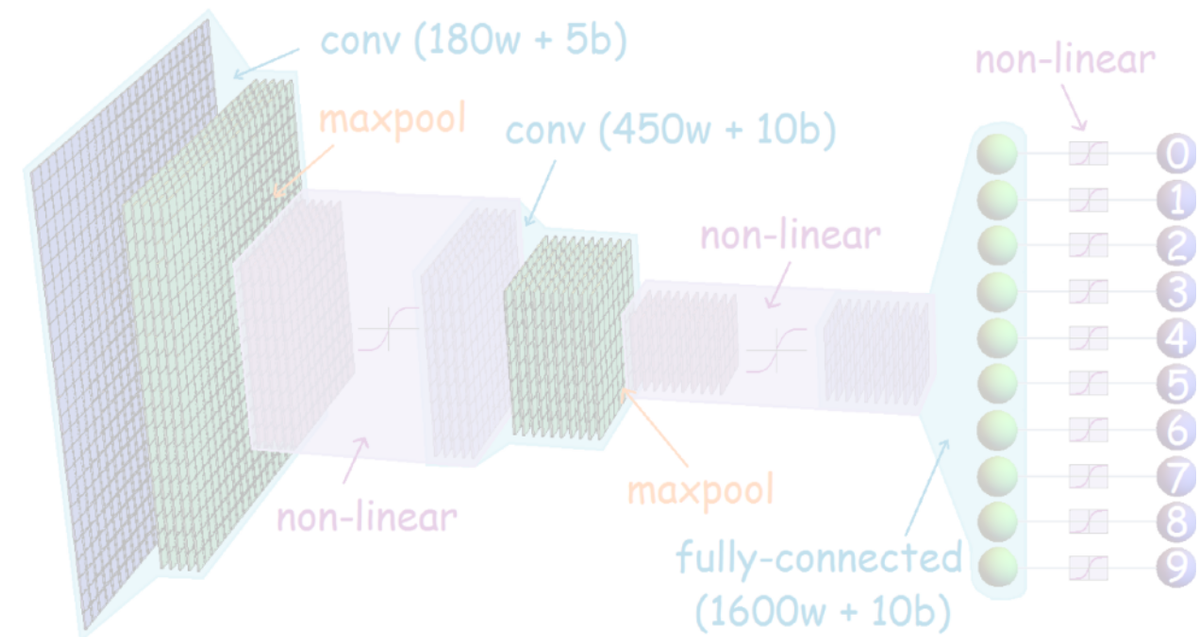
TWO APPROACHES

Use simulator
(much more efficiently)



- Approximate Bayesian Computation (ABC)
- Probabilistic Programming
- Adversarial Variational Optimization (AVO)

Learn simulator
(with deep learning)



- Generative Adversarial Networks (GANs), Variational Auto-Encoders (VAE)
- Likelihood ratio from classifiers (CARL)
- Autogregressive models, Normalizing Flows

[image credit: A.P. Goucher]

NEW! AVO

G. Louppe & K.C. arXiv:1707.07113

Adversarial Variational Optimization of Non-Differentiable Simulators

Gilles Louppe[1] and **Kyle Cranmer**[1]
[1]*New York University*

Complex computer simulators are increasingly used across fields of science as generative models tying parameters of an underlying theory to experimental observations. Inference in this setup is often difficult, as simulators rarely admit a tractable density or likelihood function. We introduce Adversarial Variational Optimization (AVO), a likelihood-free inference algorithm for fitting a non-differentiable generative model incorporating ideas from empirical Bayes and variational inference. We adapt the training procedure of generative adversarial networks by replacing the differentiable generative network with a domain-specific simulator. We solve the resulting non-differentiable minimax problem by minimizing variational upper bounds of the two adversarial objectives. Effectively, the procedure results in learning a proposal distribution over simulator parameters, such that the corresponding marginal distribution of the generated data matches the observations. We present results of the method with simulators producing both discrete and continuous data.

Leo is *G* Tom is *D*

Similar to GAN setup, but instead of using a neural network as the generator, use the actual simulation (eg. Pythia, GEANT)

Continue to use a neural network discriminator / critic.

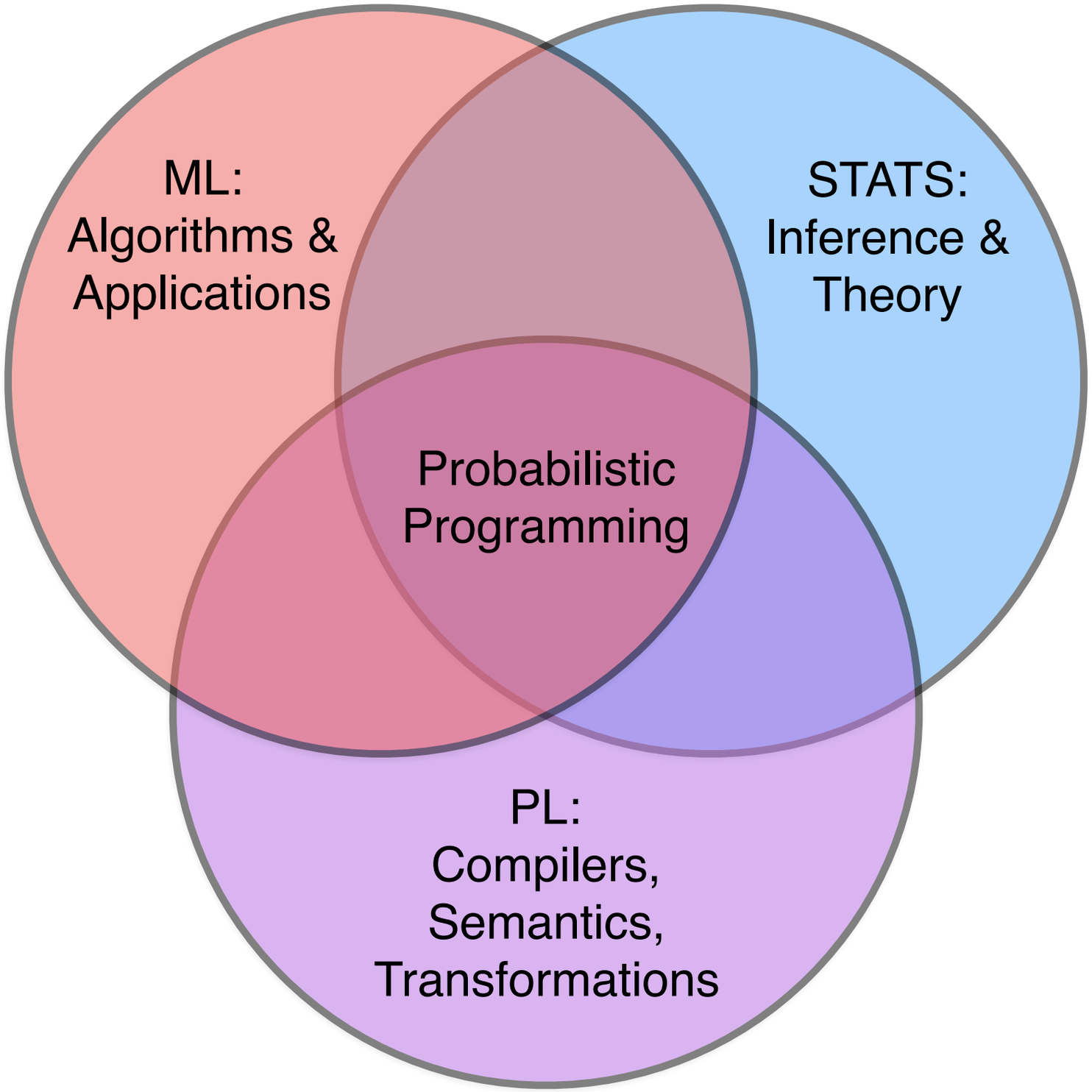
Difficulty: the simulator isn't differentiable, but there's a **trick**!

Allows us to efficiently fit / **tune simulation** with stochastic gradient techniques!

Probabilistic Programming: Inverting the simulation

(very ambitious)

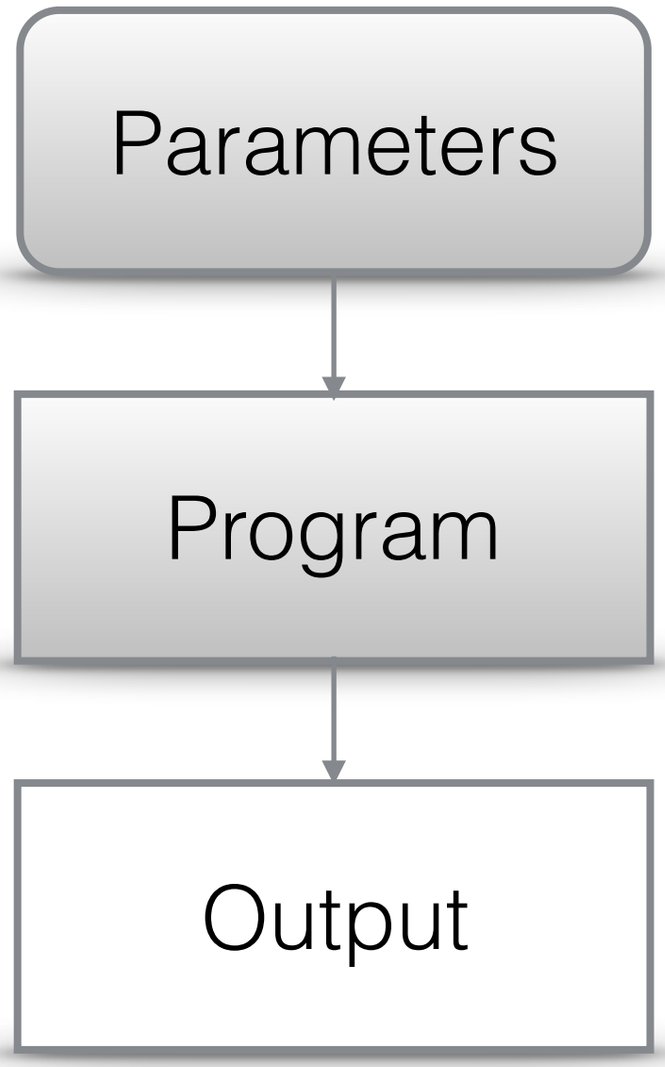
Probabilistic Programming



Intuition

[slides, Frank Wood]

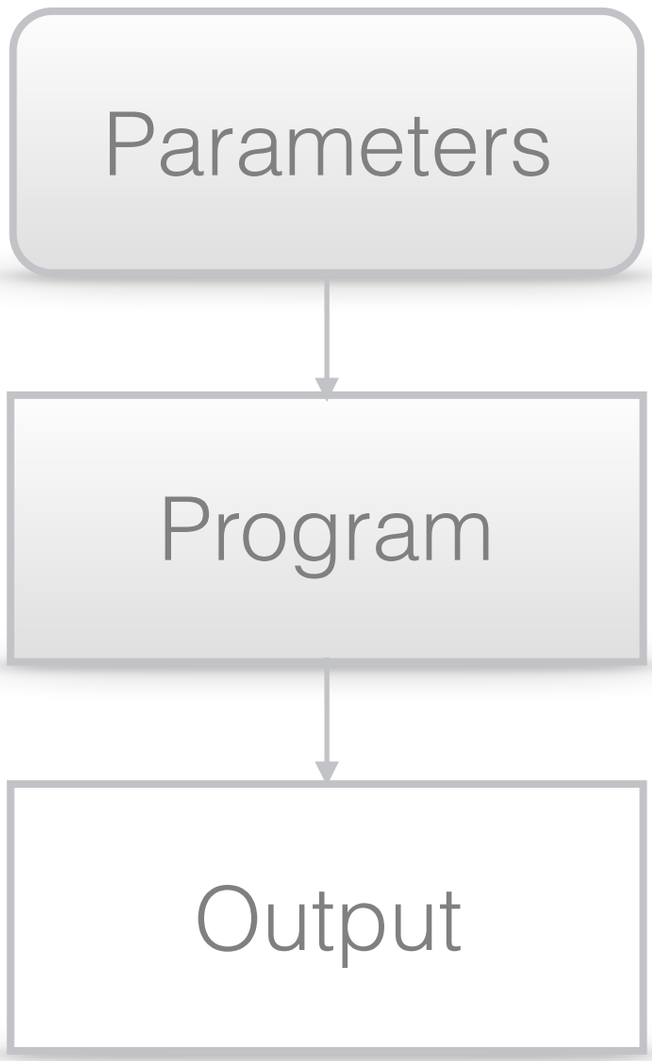
Intuition



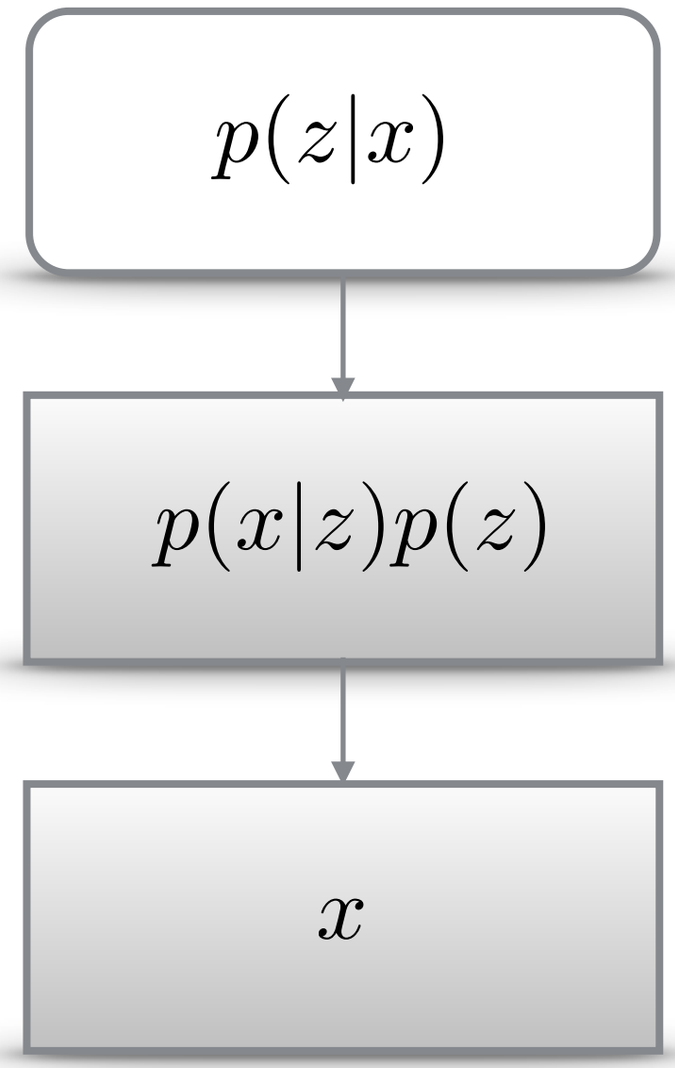
CS

[slides, Frank Wood]

Intuition



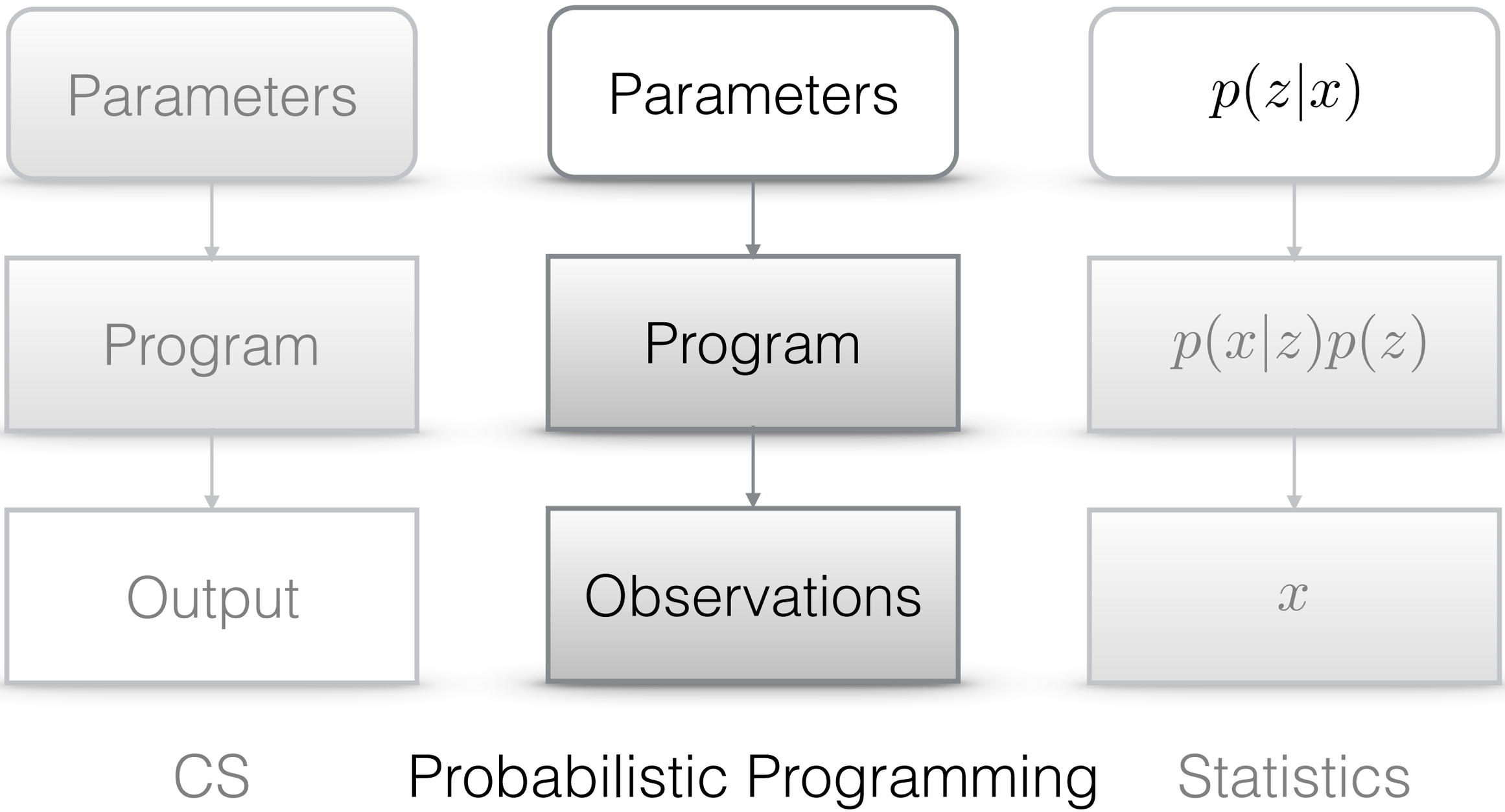
CS



Statistics

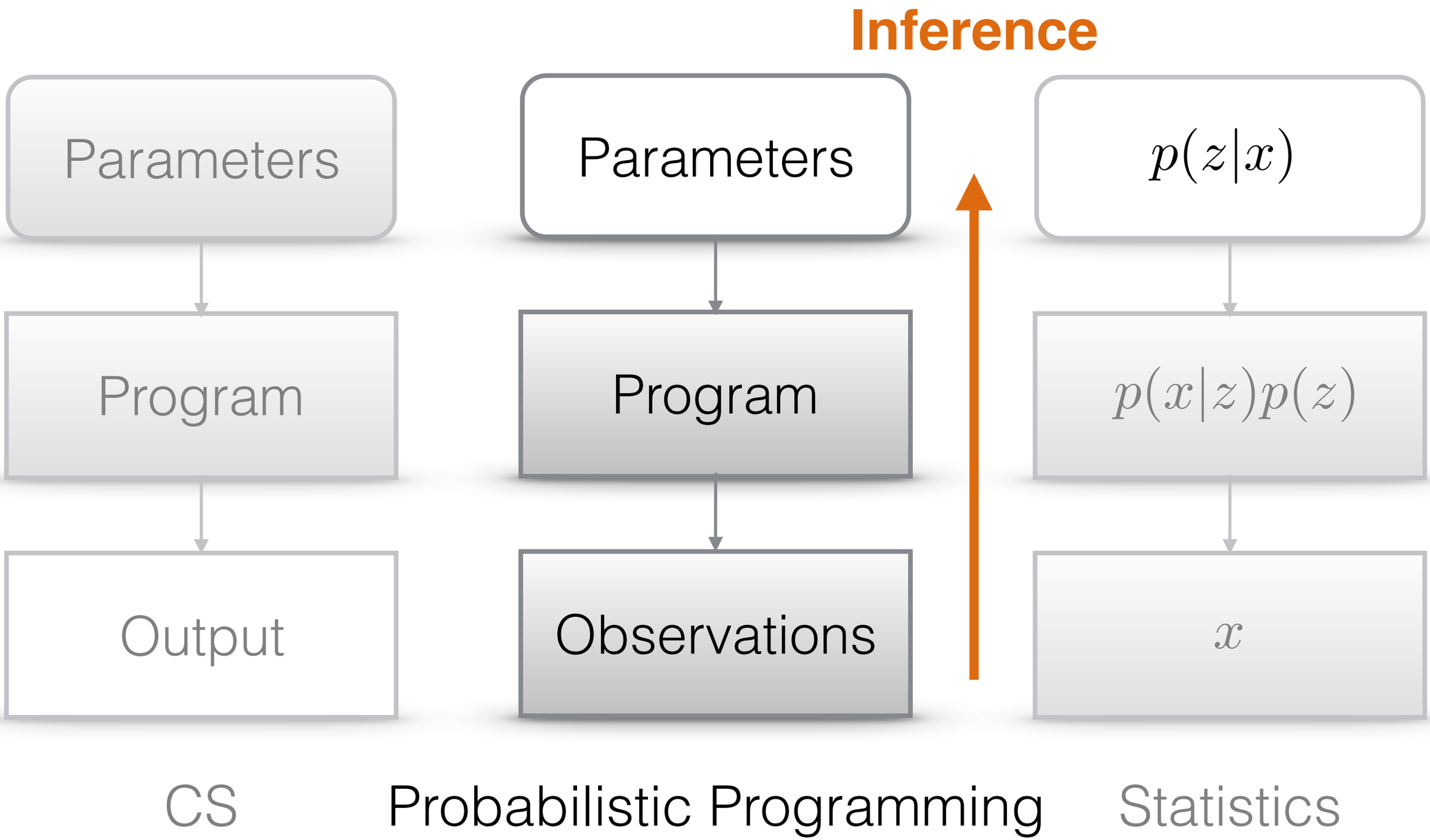
[slides, Frank Wood]

Intuition



[slides, Frank Wood]

Intuition



[slides, Frank Wood]

CAPTCHA breaking

Observation

Posterior Samples

Generative Model

```
(defquery captcha
 [image num-chars tol]
 (let [[w h] (size image)
       ;; sample random characters
       num-chars (sample
                  (poisson num-chars))
       chars (repeatedly
              num-chars sample-char)]
  ;; compare rendering to true image
  (map (fn [y z]
         (observe (normal z tol) y))
       (reduce-dim image)
       (reduce-dim (render chars w h)))
 ;; predict captcha text
 {:text
  (map :symbol (sort-by :x chars))}))
```

$\mathbf{x}$	$\mathbf{y}$
text	image

Mansinghka,, Kulkarni, Perov, and Tenenbaum
"Approximate Bayesian image interpretation using generative probabilistic graphics programs." NIPS (2013).

CAPTCHA breaking

Observation

Posterior Samples

Generative Model

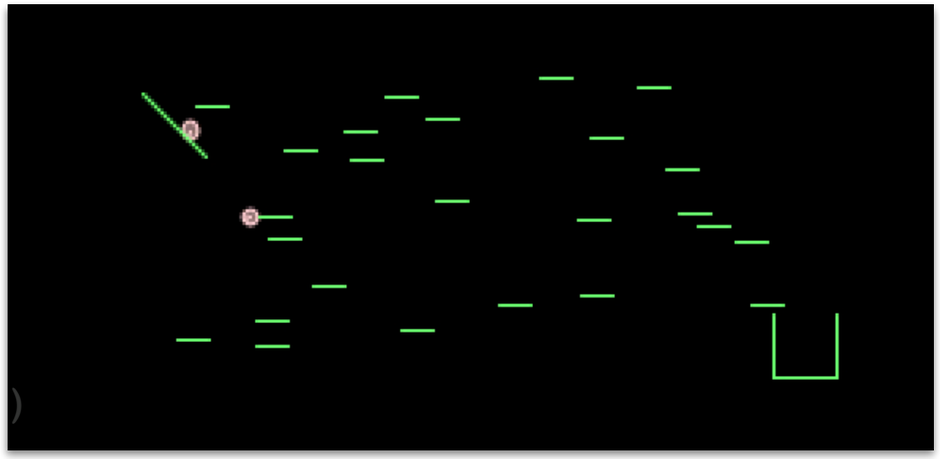
```
(defquery captcha
 [image num-chars tol]
 (let [[w h] (size image)
       ;; sample random characters
       num-chars (sample
                  (poisson num-chars))
       chars (repeatedly
              num-chars sample-char)]
  ;; compare rendering to true image
  (map (fn [y z]
         (observe (normal z tol) y))
       (reduce-dim image)
       (reduce-dim (render chars w h)))
 ;; predict captcha text
 {:text
  (map :symbol (sort-by :x chars))}))
```

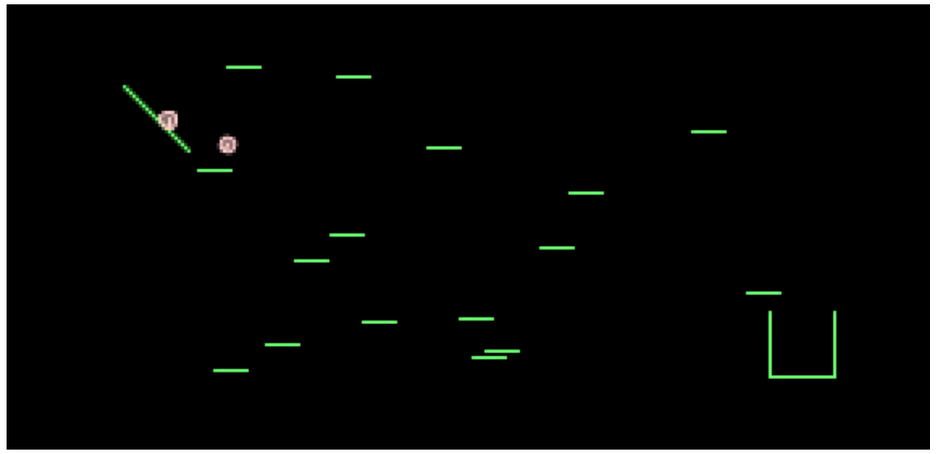
$\mathbf{x}$	$\mathbf{y}$
text	image

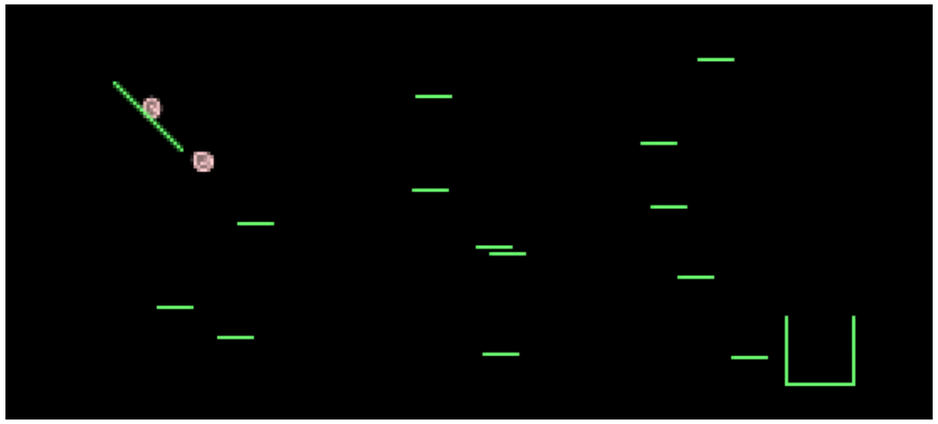
Mansinghka,, Kulkarni, Perov, and Tenenbaum
"Approximate Bayesian image interpretation using generative probabilistic graphics programs." NIPS (2013).

ANALOGY: RANDOM BUMPERS ~ RANDOM CALORIMETER SHOWER

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)]

      {:balls balls
       :num-balls-in-box num-balls-in-box
       :bumper-positions bumper-positions}))
```





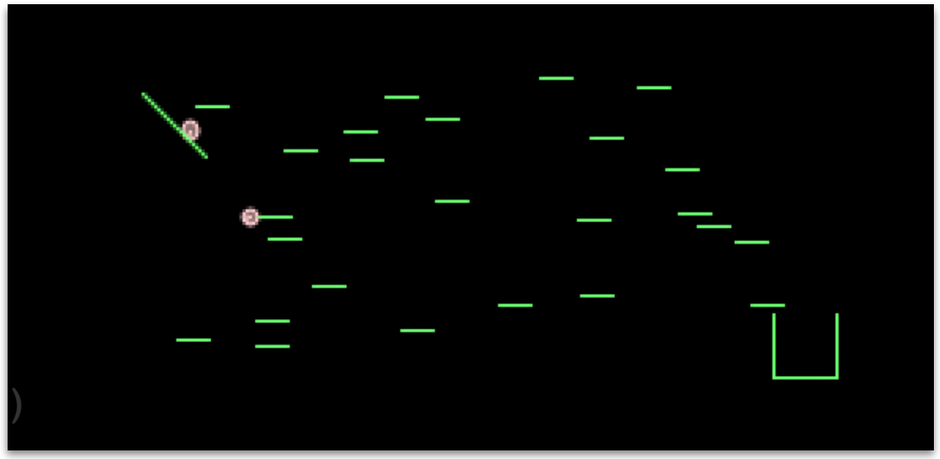


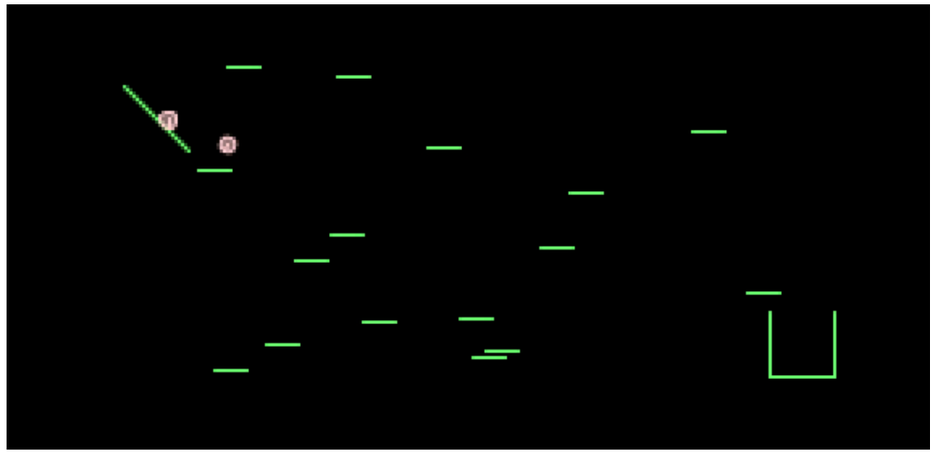
3 examples generated from simulator

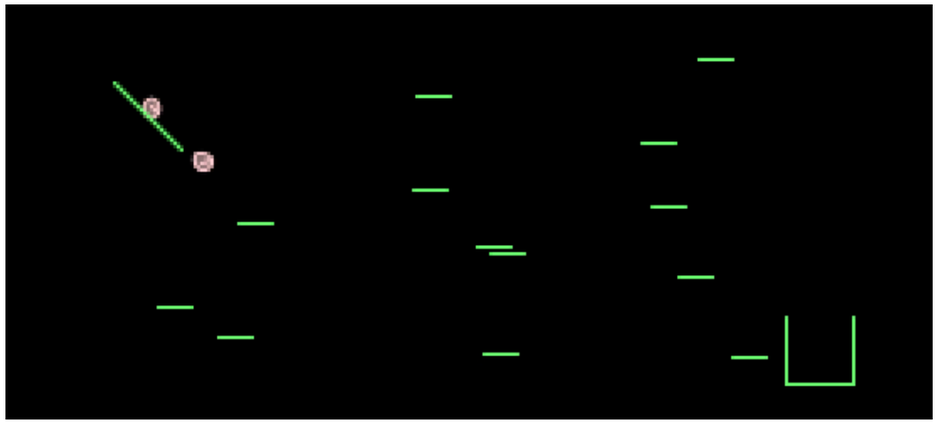
[slides, Frank Wood]

ANALOGY: RANDOM BUMPERS ~ RANDOM CALORIMETER SHOWER

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)]

      {:balls balls
       :num-balls-in-box num-balls-in-box
       :bumper-positions bumper-positions}))
```





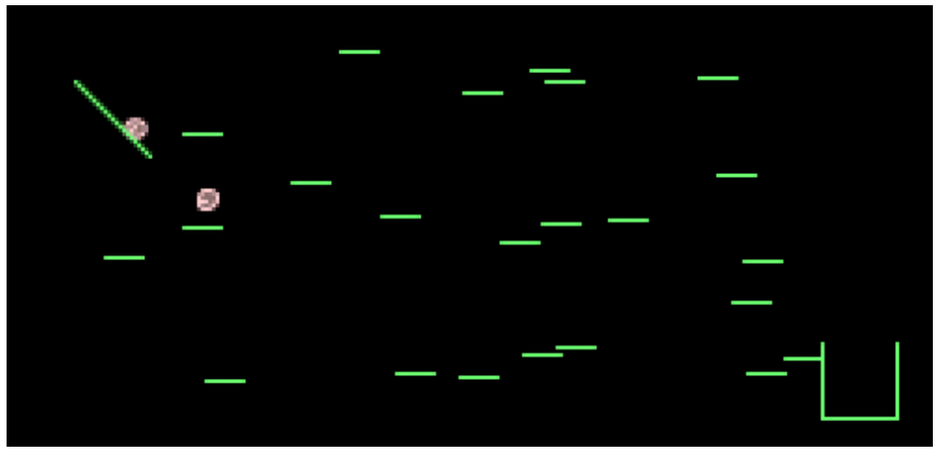


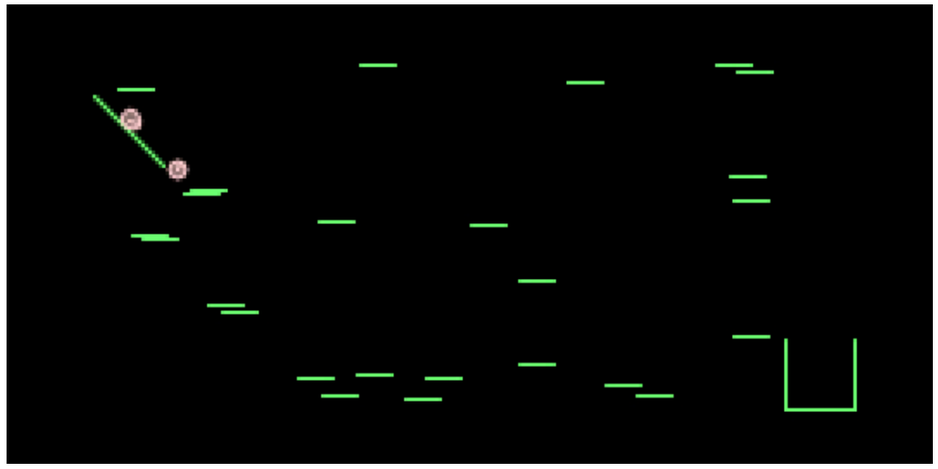
3 examples generated from simulator

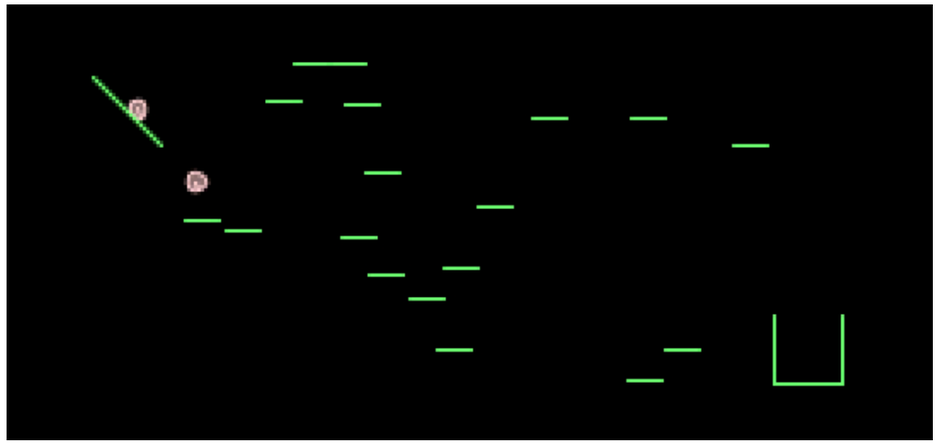
[slides, Frank Wood]

UNDERSTANDING THE TAILS OF DISTRIBUTIONS

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)

          obs-dist (normal 4 0.1)]

    (observe obs-dist num-balls-in-box)
```





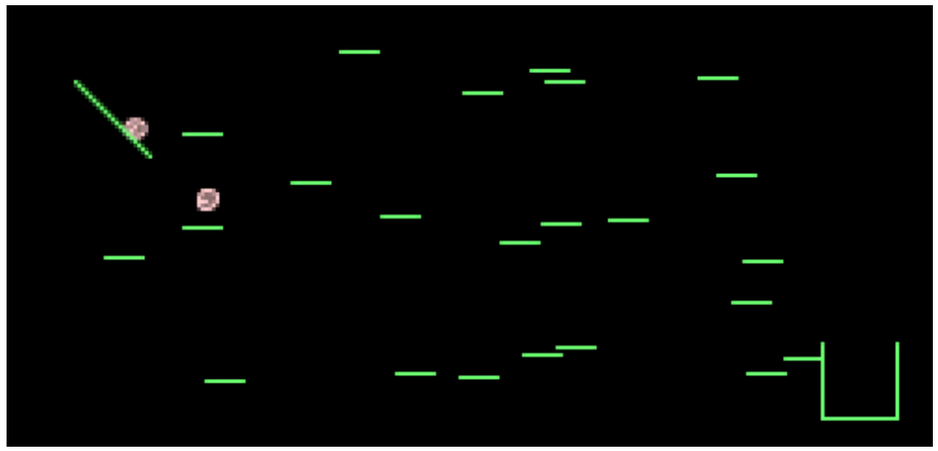


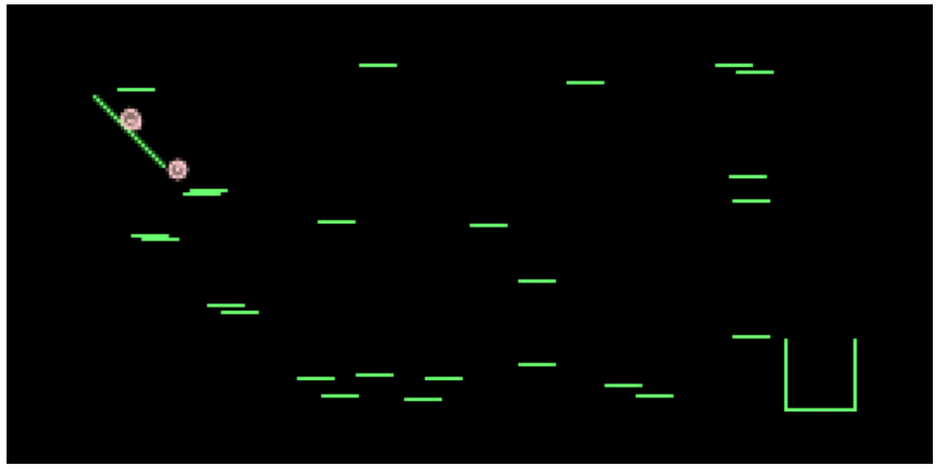
3 examples generated from simulator **conditioned** on ~20% of balls land in box (~ given observed energy deposits)

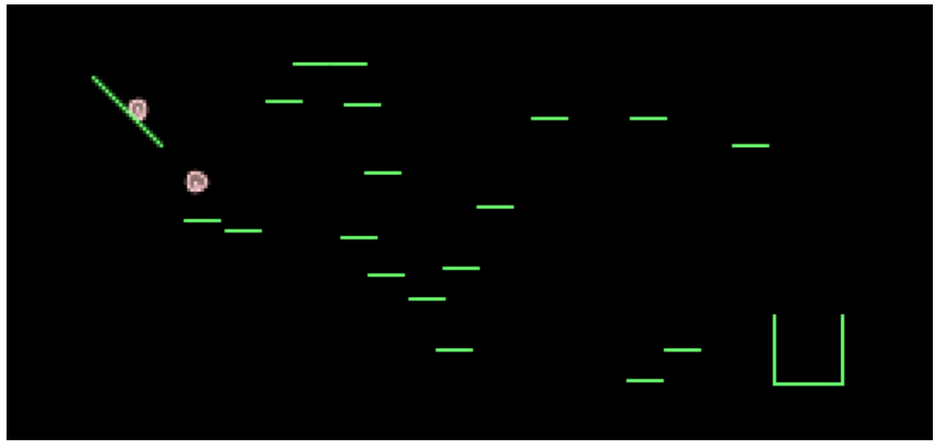
[slides, Frank Wood]

UNDERSTANDING THE TAILS OF DISTRIBUTIONS

```
(defquery arrange-bumpers []
    (let [number-of-bumpers (sample (poisson 20))
          bumpydist (uniform-continuous 0 10)
          bumpxdist (uniform-continuous -5 14)
          bumper-positions (repeatedly
                            number-of-bumpers
                            #(vector (sample bumpxdist)
                                     (sample bumpydist)))

          ;; code to simulate the world
          world (create-world bumper-positions)
          end-world (simulate-world world)
          balls (:balls end-world)

          ;; how many balls entered the box?
          num-balls-in-box (balls-in-box end-world)

          obs-dist (normal 4 0.1)]

    (observe obs-dist num-balls-in-box)
```







3 examples generated from simulator **conditioned** on ~20% of balls land in box (~ given observed energy deposits)

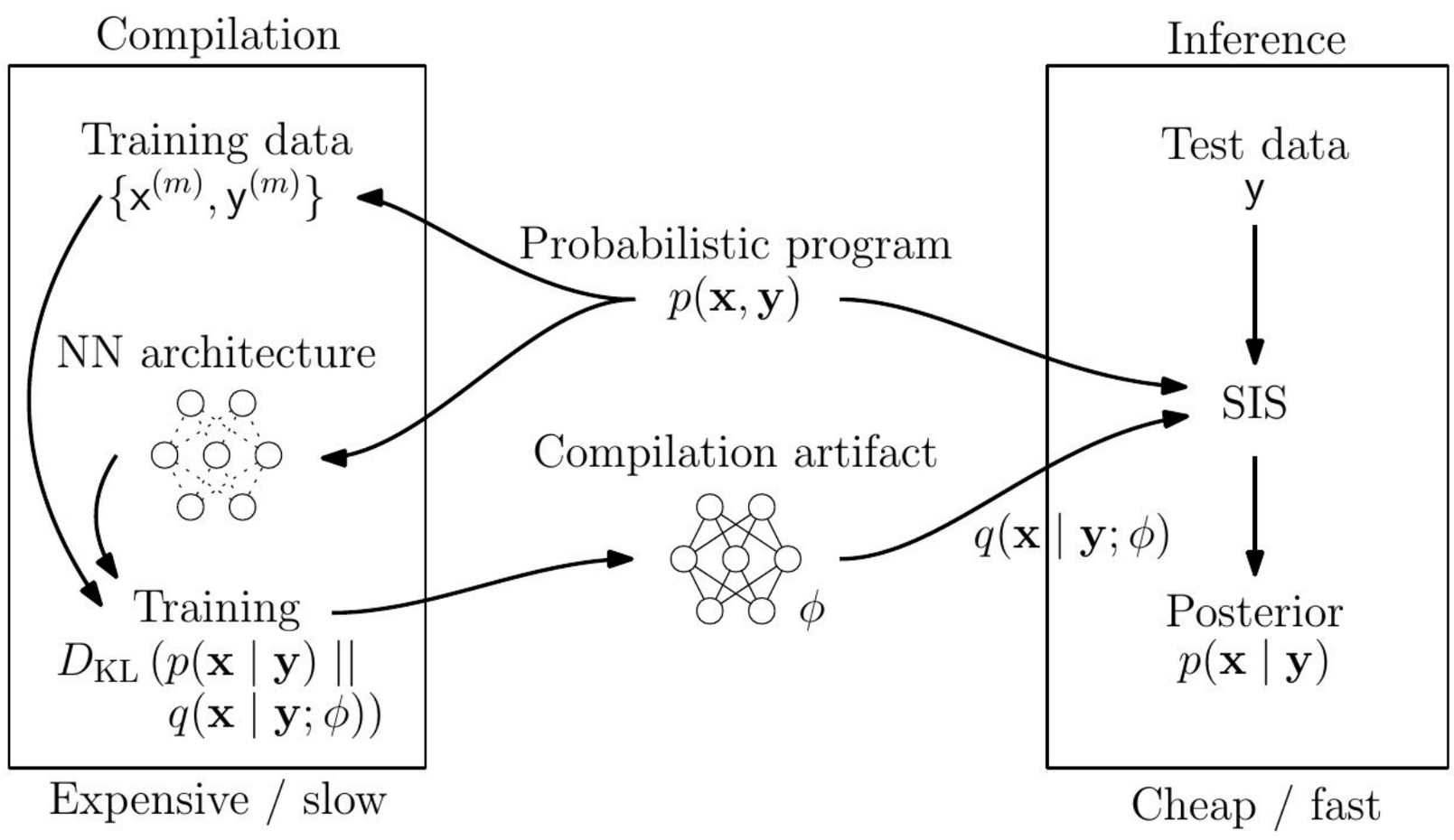
[slides, Frank Wood]

HOW DOES IT WORK?

In short: hijack the random number generators and use NN's to perform a *very* smart type of importance sampling

Input: an inference problem denoted in a universal PPL (Anglican, CPProb)

Output: a trained inference network, or "compilation artifact" (Torch, PyTorch)



Le, Baydin and Wood. Inference Compilation and Universal Probabilistic Programming. AISTATS 2017. *arXiv:1610.09900*

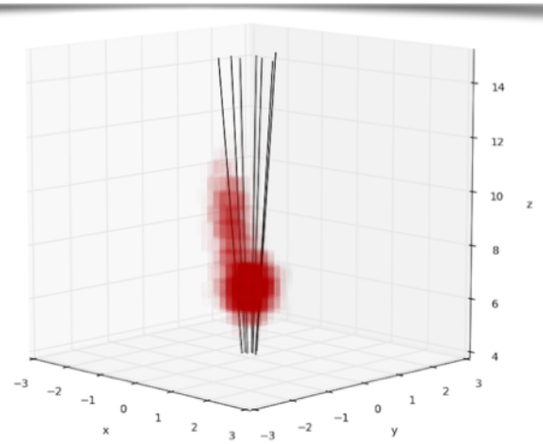
IN PROGRESS: C++, SHERPA, GEANT4

Mario Lezcano Casado, Atılım Güneş Baydin, Tuan Anh Le, Frank Wood*
Department of Engineering Science
University of Oxford
{lezcano,gunes,tuananh,fwood}@robots.ox.ac.uk

Lukas Heinrich, Gilles Louppe, Kyle Cranmer
Department of Physics & Center for Data Science
New York University
{kyle.cranmer,lukas.heinrich,g.louppe}@cern.ch

Wahid Bhimji, Prabhat
Lawrence Berkeley National Laboratory
{wbhimji,prabhat}@lbl.gov

Karen Ng
Intel
karen.y.ng@intel.com



Probabilistic programming with C++

Our new tool: CPProb
https://github.com/probprog/cpprob

Instrumenting C++ code to allow tools like SHERPA and GEANT run with inference compilation

```
void linear_regression(const std::array<std::pair<RealType, RealType>, N> & points) {
    using boost::random::normal_distribution;

    auto normal = normal_distribution<RealType>{0, 10};
    const auto a = cpprob::sample(normal, true);
    const auto b = cpprob::sample(normal, true);

    for (const auto & point : points) {
        auto likelihood = normal_distribution<RealType>{a * point.first + b, 1};
        cpprob::observe(likelihood, point.second);
    }
    cpprob::predict(a);
    cpprob::predict(b);
}
```

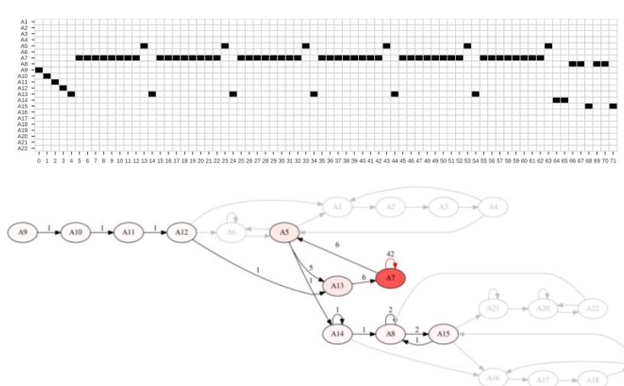
```
SHERPA::Hadron_Decays::Treat(ATOOLS::Blob_List*, double&)+0x709
SHERPA::Event_Handler::IterateEventPhases(SHERPA::eventtype::code&, double&)+0x1b2
SHERPA::Event_Handler::GenerateHadronDecayEvent(SHERPA::eventtype::code&)+0x979
```

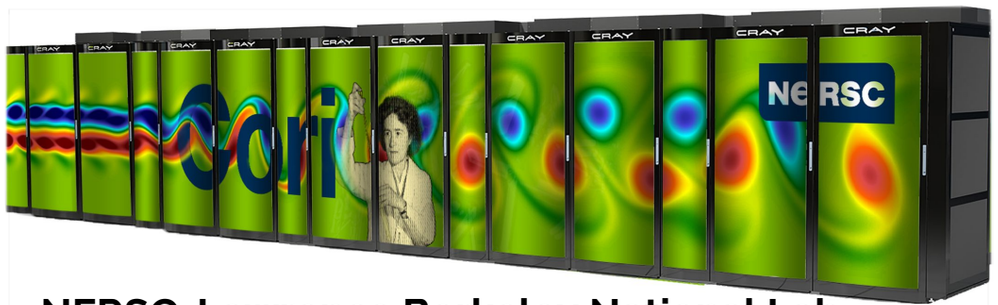
A case study in SHERPA & GEANT

Probabilistic program analytics
allows us to **pinpoint "interesting" addresses** in execution traces
and corresponding **C++ code within SHERPA**

4.4.24 Unique trace T24

Length 72





NERSC, Lawrence Berkeley National Lab

Our current tools:
- CPProb
  - A new C++ PPL coupled with large-scale simulations using, e.g., SHERPA and GEANT
- PyTorch inference compilation backend
  - Dynamic computation graphs for NN artifacts

Designed to run on Cori at NERSC using Shifter

```
shifterimg -v pull docker:gbaydin/pytorch-infcomp:latest
shifterimg -v pull docker:gbaydin/sherpa-infcomp-full:latest
```

Slides from Atılım Güneş Baydin @ Hammers & Nails

Active Learning

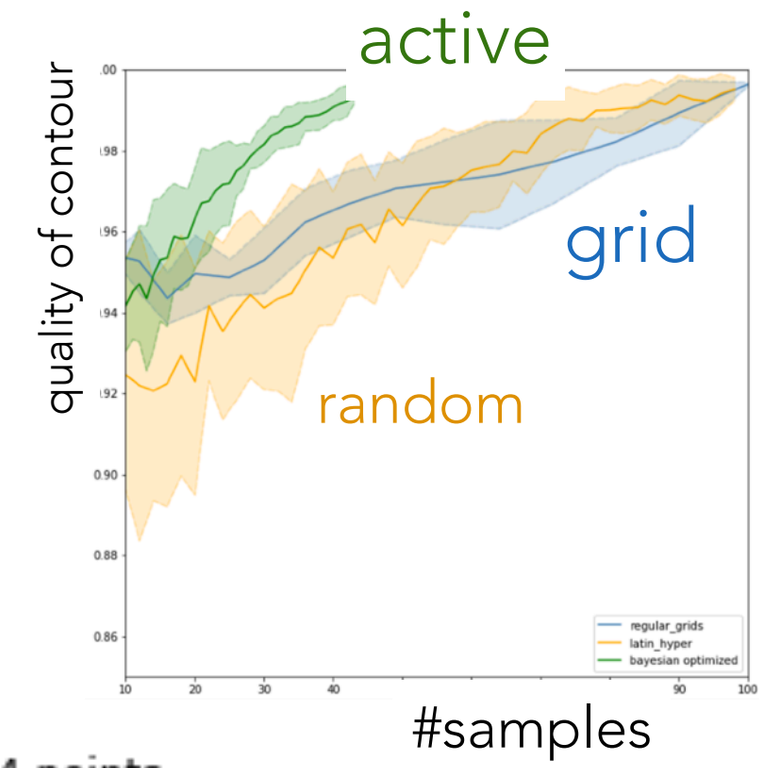
more efficient simulation

ACTIVE LEARNING

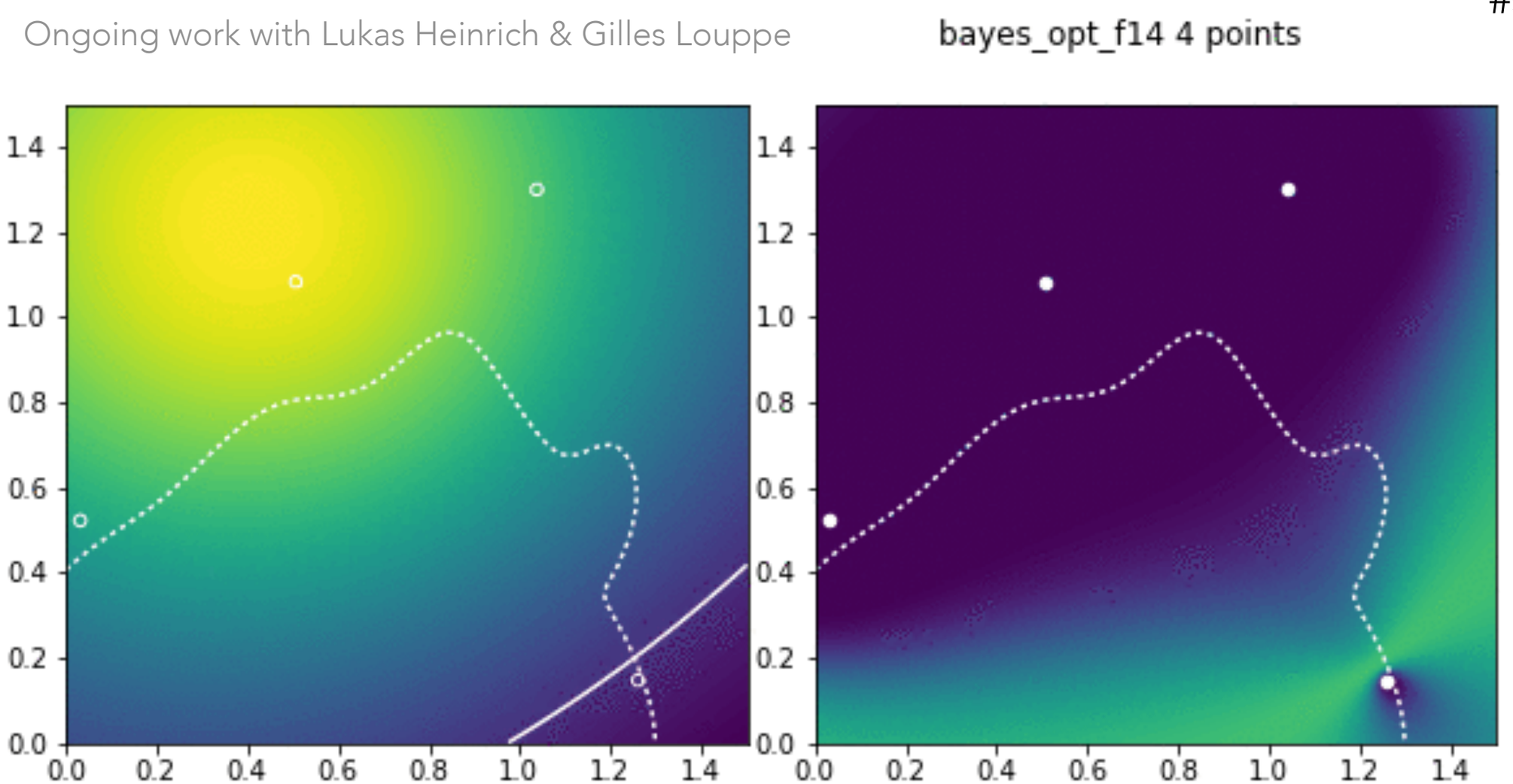
Instead of generating Monte Carlo a priori, generate it on demand where it is relevant!

↓ An algorithm for finding exclusion contours

Drastically more efficient use of computing resources →



Ongoing work with Lukas Heinrich & Gilles Louppe

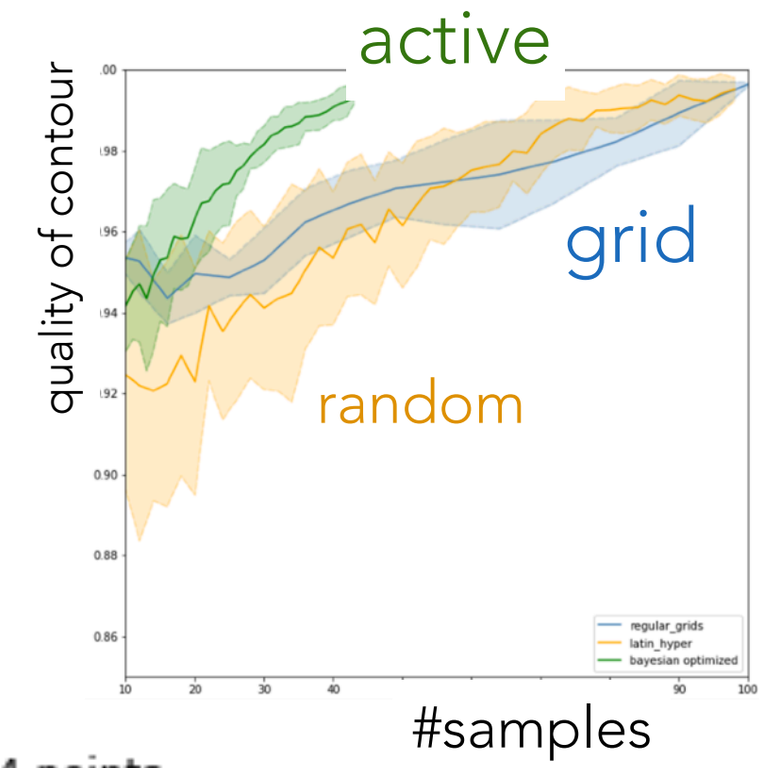


ACTIVE LEARNING

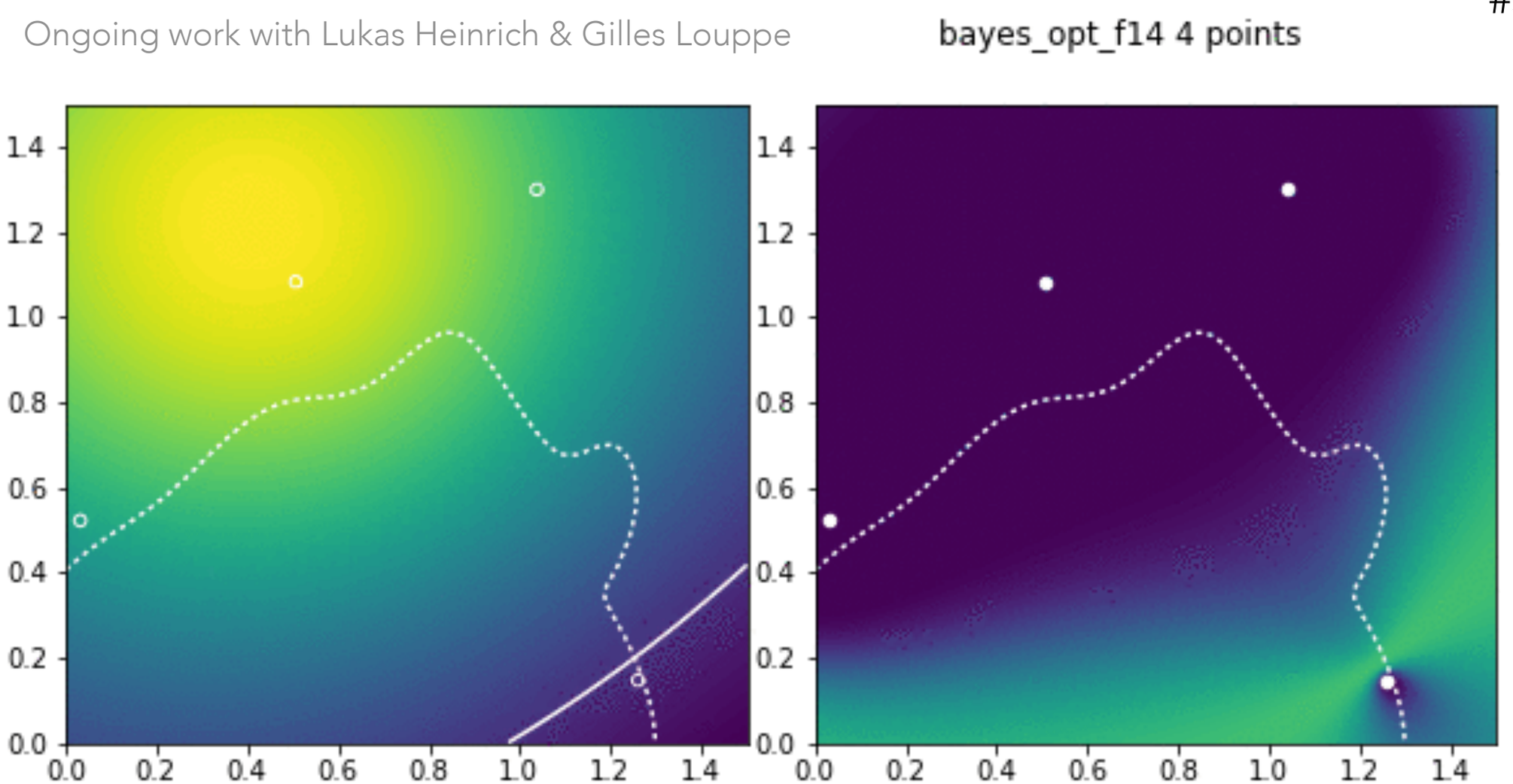
Instead of generating Monte Carlo a priori, generate it on demand where it is relevant!

↓ An algorithm for finding exclusion contours

Drastically more efficient use of computing resources →



Ongoing work with Lukas Heinrich & Gilles Louppe



SYNTHESIS

active learning

Active Sciencing

reusable workflows

simulation-based
inference engines

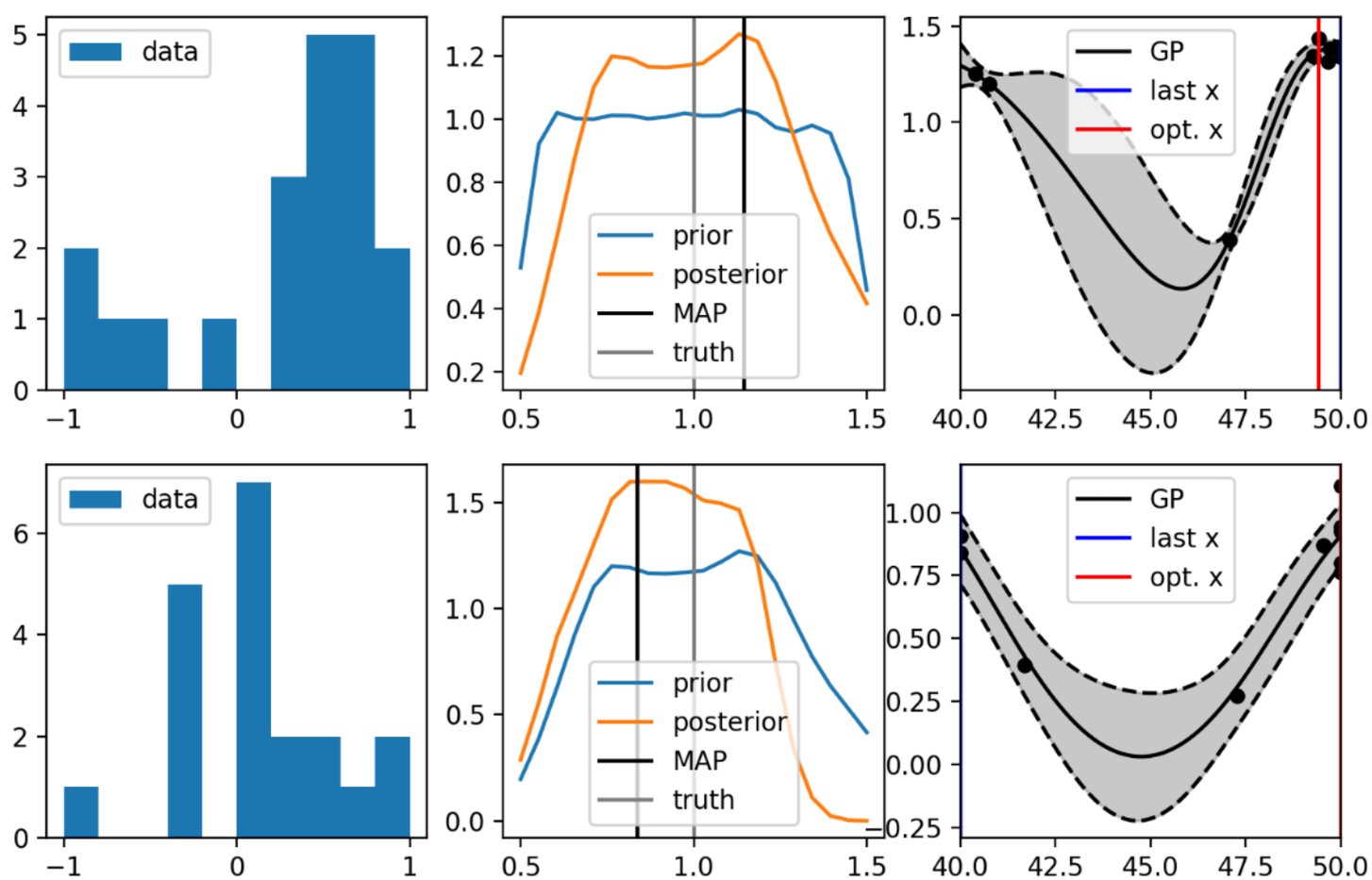
https://github.com/cranmer/active_sciencing

ACTIVE SCIENCING DEMO

Input:

- workflow for performing "real" experiment that returns data
- workflow for running simulator given parameters of theory and experimental configuration

Demo shows use of likelihood-free inference technique & Bayesian Optimization to measure the Weinberg angle and optimize beam energy (eg. just above or below $M_Z/2$)



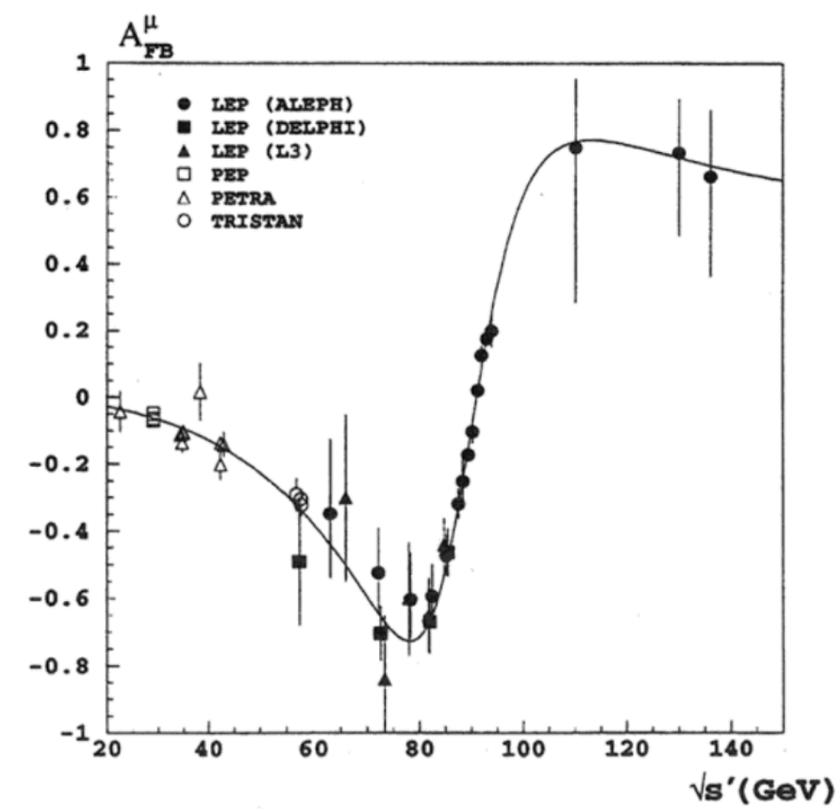


Figure 2: Measured forward-backward asymmetries of muon-pair production compared with the model independent fit results.

CONCLUSIONS

The developments in machine learning and AI go way beyond improved classifiers and have the potential to revolutionize high energy & nuclear physics

- generative models and likelihood-free inference are two particularly exciting areas

Our understanding of how to leverage our prior physics knowledge while letting machine learning do what it's good at is maturing.

- exploit hierarchical structure of events
- individually validated & reusable components

Harnessing the full potential of these techniques will require deep integration into our software and changes to our computing models

Backup / Reference

THE CRUX, AN INTRACTABLE INTEGRAL

observed

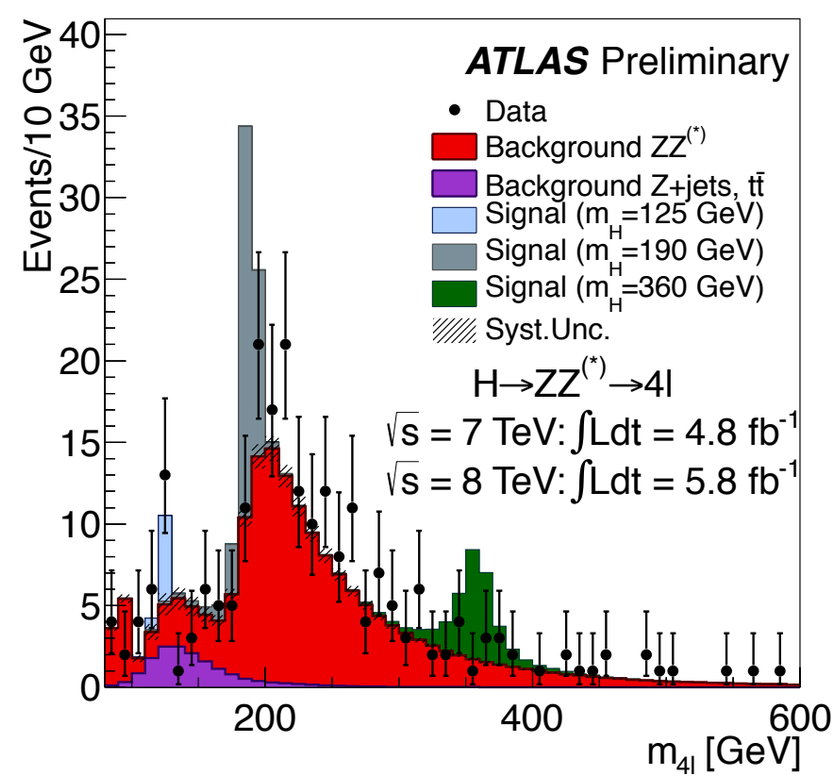
Monte Carlo Sampling

MC Truth

$$p(x|\theta) = \int dz\, p(x, z|\theta)$$

$$\hat{p}(x|\theta)$$

histogram approximation



‘Likelihood-Free’ Inference

← exact Bayesian Computation

Rejection Algorithm

- Draw θ from prior $\pi(\cdot)$
- Accept θ with probability $\pi(D \mid \theta)$

Accepted θ are independent draws from the posterior distribution, $\pi(\theta \mid D)$.

If the likelihood, $\pi(D|\theta)$, is unknown:

‘Mechanical’ Rejection Algorithm

- Draw θ from $\pi(\cdot)$
- Simulate $X \sim f(\theta)$ from the computer model
- Accept θ if $D = X$, i.e., if computer output equals observation

The acceptance rate is $\int \mathbb{P}(D|\theta)\pi(\theta)\mathrm{d}\theta = \mathbb{P}(D)$.

*From Richard Wilkinson’s talk at Data science @LHC

Rejection ABC

If $\mathbb{P}(D)$ is small (or D continuous), we will rarely accept any θ. Instead, there is an approximate version:

Uniform Rejection Algorithm

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(D, X) \leq \epsilon$

ϵ reflects the tension between computability and accuracy.

- As $\epsilon \to \infty$, we get observations from the prior, $\pi(\theta)$.
- If $\epsilon = 0$, we generate observations from $\pi(\theta \mid D)$.

For reasons that will become clear later, we call this *uniform-ABC*.

*From Richard Wilkinson's talk at Data science @LHC

CARL SOFTWARE

http://diana-hep.org/carl/

diana-hep.org

DiscoveryLinks Higgs RooStats ALEPH Apple News Life Stuff ATLAS Wikipedia, inSpire Theory&Practice nyu espace JCSS HCG

Meet F | Jupyter Note... | Weekend rea... | early-career-... | 2016 Electio... | 12-day Event... | Joint meetin... | carl API doc...

Fork me on GitHub

Index

Sub-modules

- carl.data
- carl.distributions
- carl.learning
- carl.ratios

Notebooks

- Composing and fitting distributions
- Diagnostics for approximate likelihood ratios
- Likelihood ratios of mixtures of normals
- Parameterized inference from multidimensional data
- Parameterized inference with nuisance parameters

Display a menu

carl module

`carl` is a toolbox for likelihood-free inference in Python.

The likelihood function is the central object that summarizes the information from an experiment needed for inference of model parameters. It is key to many areas of science that report the results of classical hypothesis tests or confidence intervals using the (generalized or profile) likelihood ratio as a test statistic. At the same time, with the advance of computing technology, it has become increasingly common that a simulator (or generative model) is used to describe complex processes that tie parameters of an underlying theory and measurement apparatus to high-dimensional observations. However, directly evaluating the likelihood function in these cases is often impossible or is computationally impractical.

In this context, the goal of this package is to provide tools for the likelihood-free setup, including likelihood (or density) ratio estimation algorithms, along with helpers to carry out inference on top of these.

This project is still in its early stage of development. Join us on GitHub if you feel like contributing!

build passing | coverage 91% | DOI 10.5281/zenodo.47798

Likelihood-free inference with calibrated classifiers

Extensive details regarding likelihood-free inference with calibrated classifiers can be found in the companion paper *"Approximating Likelihood Ratios with Calibrated Discriminative Classifiers", Kyle Cranmer, Juan Pavez, Gilles Louppe.* http://arxiv.org/abs/1506.02169

Installation

The following dependencies are required:

- Numpy >= 1.11

For CARL: see K.C., G. Louppe, J. Pavez: http://arxiv.org/abs/1506.02169

beta.briefideas.org

DiscoveryLinks · Higgs · RooStats · ALEPH · Apple · News · Life Stuff · ATLAS · Wikipedia, · inSpire · Theory&Practice · nyu espace · JCSS · HCG · Evernote

Reconstituting Asteroids into Mechanical Automata | NASA | The Journal of Brief Ideas

Unifying generative models and exact likelihood-free inference with conditional bijections

By *Kyle Cranmer*, *Gilles Louppe*

machine learning | likelihood-free inference | density estimation

Recent work in density estimation uses a bijection $f : X \to Z$ (e.g. an invertible flow or autoregressive model) and a tractable density $p(z)$ (e.g. [1] [2] [3] [4]).

$$p(x) = p(f_\phi(x)) \left| \det \left(\frac{\partial f_\phi(x)}{\partial x_T} \right) \right| ,$$

where ϕ are the internal network parameters for the bijection f_ϕ. Learning proceeds via gradient ascent $\nabla_\phi \sum_i \log p(x_i)$ with data x_i (i.e. maximum likelihood wrt. the internal parameters ϕ). Since f is invertible, then this model can also be used as a generative model for X.

This can be generalized to the conditional density $p(x|\theta)$ by utilizing a family of bijections $f_\theta : X \to Z$ parametrized by θ (e.g. [5] [6]).

$$p(x|\theta) = p(f_{\phi;\theta}(x)) \left| \det \left(\frac{\partial f_{\phi;\theta}(x)}{\partial x_T} \right) \right|$$

Here θ and x are input to the network (and its inverse) and ϕ are internal network parameters. Again, learning proceeds via gradient ascent $\nabla_\phi \sum_i \log p(x_i|\theta_i)$ with data x_i, θ_i.

We observe that not only can this model be used as a conditional generative model $p(x|\theta)$, but it can also be used to perform asymptotically exact, amortized likelihood-free inference on θ.

This is particularly interesting when θ is identified with the parameters of an intractable, non-differentiable computer simulation or the conditions of some real world data collection process.

Comments

Many thanks to Durk Kingma, Max Welling, Ian Goodfellow, and Shakir Mohamed for enlightening discussions at NIPS2016.

Kyle Cranmer · 9 Dec, 2016

Kyle Cranmer · Sign out

Actions

1 vote | Hide | Collect

Share 111 | Tweet 11

Authors

Kyle Cranmer, Gilles Louppe

Metadata

DOI https://doi.org/10.5281/zenodo.198541

Published: 8 Dec, 2016

Display a menu

BAYES OPT IN A NUTSHELL

[slides from Gilles Louppe]

Bayesian optimisation

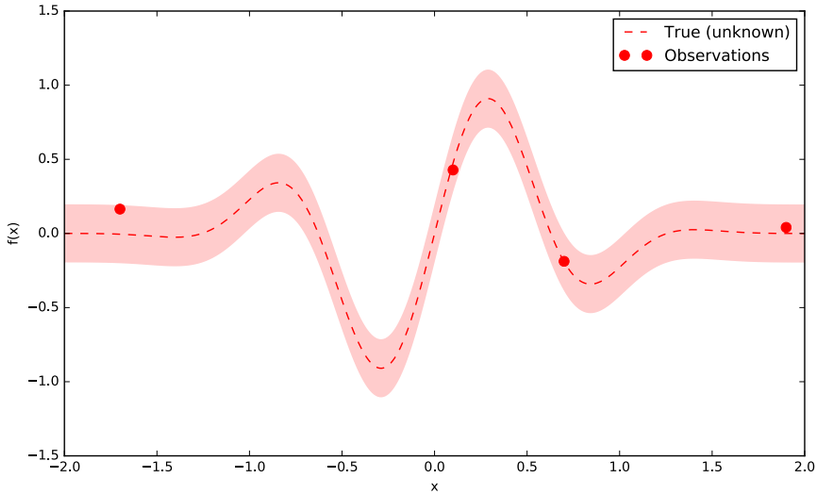
for $t = 1 : T$,

1. Given observations (x_i, y_i) for $i = 1 : t$, build a probabilistic model for the objective f.
   - Integrate out all possible true functions, using Gaussian process regression.
2. Optimise a cheap utility function u based on the posterior distribution for sampling the next point.

$$x_{t+1} = \arg\max_x u(x)$$

   Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

Where shall we sample next?



BAYES OPT IN A NUTSHELL

[slides from Gilles Louppe]

Bayesian optimisation

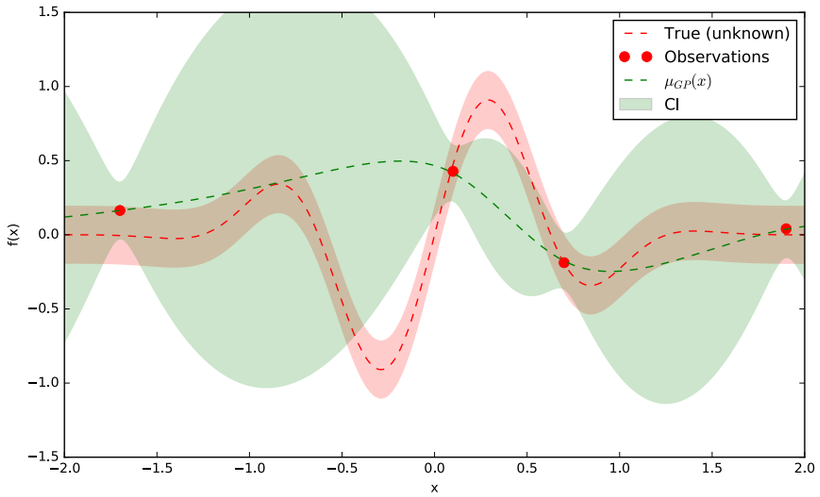
for $t = 1 : T$,

1. Given observations (x_i, y_i) for $i = 1 : t$, build a probabilistic model for the objective f.
   - Integrate out all possible true functions, using Gaussian process regression.
2. Optimise a cheap utility function u based on the posterior distribution for sampling the next point.

$$x_{t+1} = \arg\max_x u(x)$$

   Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

Build a probabilistic model for the objective function



This gives a posterior distribution over functions that could have generated the observed data.

BAYES OPT IN A NUTSHELL

[slides from Gilles Louppe]

Bayesian optimisation

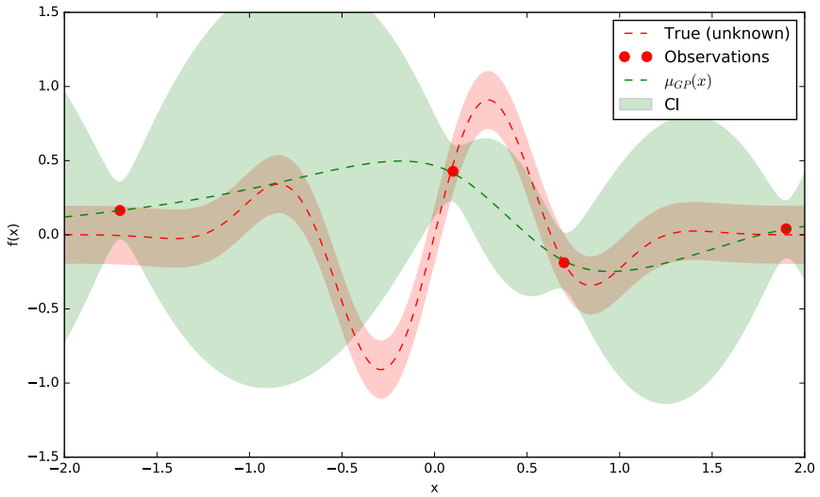
for $t = 1 : T$,

1. Given observations (x_i, y_i) for $i = 1 : t$, build a probabilistic model for the objective f.
   - Integrate out all possible true functions, using Gaussian process regression.
2. Optimise a cheap utility function u based on the posterior distribution for sampling the next point.

$$x_{t+1} = \arg\max_x u(x)$$

   Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

Build a probabilistic model for the objective function



This gives a posterior distribution over functions that could have generated the observed data.

Acquisition functions

Acquisition functions u(x) specify which sample x should be tried next:

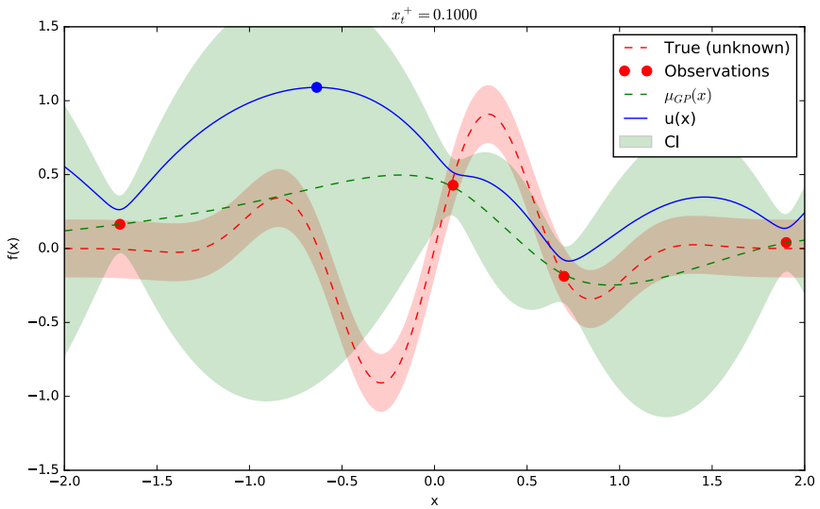
- Upper confidence bound $\text{UCB}(x) = \mu_{GP}(x) + \kappa\sigma_{GP}(x)$;
- Probability of improvement $\text{PI}(x) = P(f(x) \geq f(x_t^+) + \kappa)$;
- Expected improvement $\text{EI}(x) = \mathbb{E}[f(x) - f(x_t^+)]$;
- ... and many others.

where x_t^+ is the best point observed so far.

In most cases, acquisition functions provide knobs (e.g., κ) for controlling the exploration-exploitation trade-off.

- Search in regions where $\mu_{GP}(x)$ is high (exploitation)
- Probe regions where uncertainty $\sigma_{GP}(x)$ is high (exploration)

Plugging everything together ($t = 0$)



$x_{t+1} = \arg\max_x \text{UCB}(x)$

BAYES OPT IN A NUTSHELL

[slides from Gilles Louppe]

Bayesian optimisation

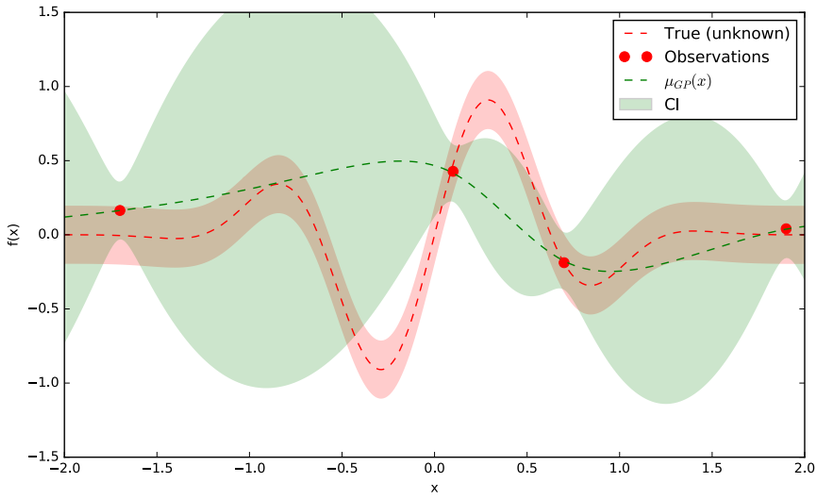
for $t = 1 : T$,

1. Given observations (x_i, y_i) for $i = 1 : t$, build a probabilistic model for the objective f.
   - Integrate out all possible true functions, using Gaussian process regression.
2. Optimise a cheap utility function u based on the posterior distribution for sampling the next point.

$$x_{t+1} = \arg\max_x u(x)$$

   Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

Build a probabilistic model for the objective function



This gives a posterior distribution over functions that could have generated the observed data.

Acquisition functions

Acquisition functions u(x) specify which sample x should be tried next:

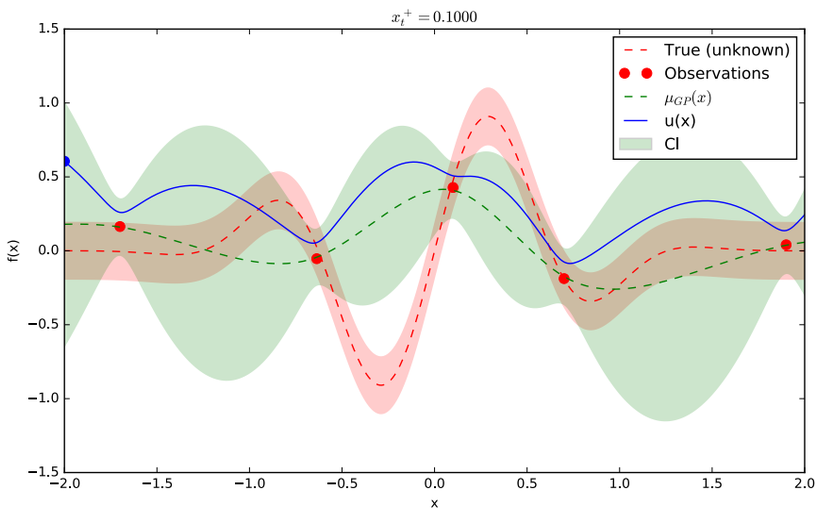
- Upper confidence bound $\text{UCB}(x) = \mu_{GP}(x) + \kappa\sigma_{GP}(x)$;
- Probability of improvement $\text{PI}(x) = P(f(x) \geq f(x_t^+) + \kappa)$;
- Expected improvement $\text{EI}(x) = \mathbb{E}[f(x) - f(x_t^+)]$;
- ... and many others.

where x_t^+ is the best point observed so far.

In most cases, acquisition functions provide knobs (e.g., κ) for controlling the exploration-exploitation trade-off.

- Search in regions where $\mu_{GP}(x)$ is high (exploitation)
- Probe regions where uncertainty $\sigma_{GP}(x)$ is high (exploration)

... and repeat until convergence ($t = 1$)



BAYES OPT IN A NUTSHELL

[slides from Gilles Louppe]

Bayesian optimisation

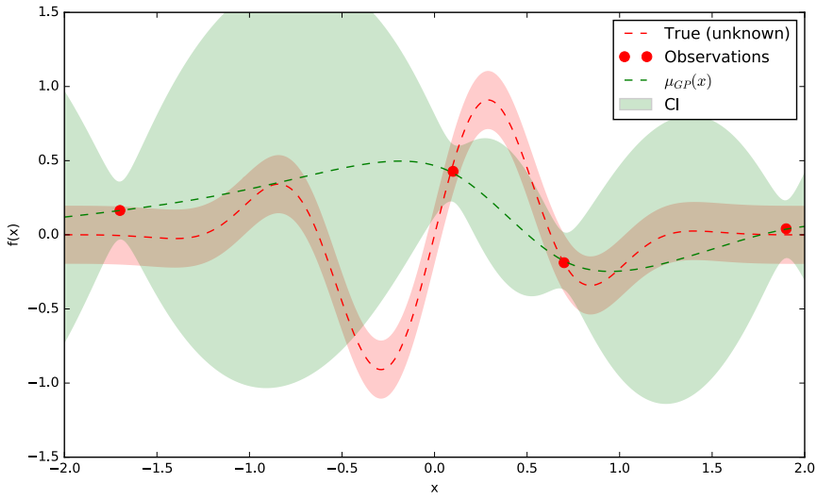
for $t = 1 : T$,

1. Given observations (x_i, y_i) for $i = 1 : t$, build a probabilistic model for the objective f.
   - Integrate out all possible true functions, using Gaussian process regression.
2. Optimise a cheap utility function u based on the posterior distribution for sampling the next point.

$$x_{t+1} = \arg\max_x u(x)$$

   Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

Build a probabilistic model for the objective function



This gives a posterior distribution over functions that could have generated the observed data.

Acquisition functions

Acquisition functions u(x) specify which sample x should be tried next:

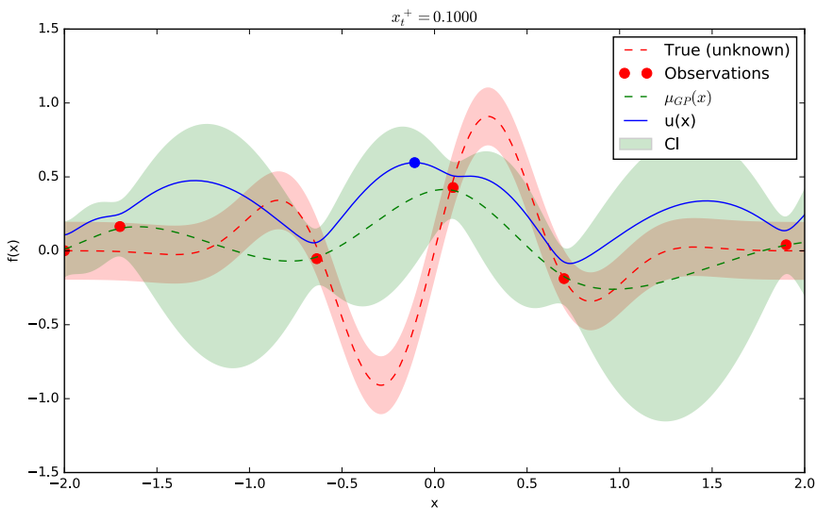
- Upper confidence bound $\text{UCB}(x) = \mu_{GP}(x) + \kappa\sigma_{GP}(x)$;
- Probability of improvement $\text{PI}(x) = P(f(x) \geq f(x_t^+) + \kappa)$;
- Expected improvement $\text{EI}(x) = \mathbb{E}[f(x) - f(x_t^+)]$;
- ... and many others.

where x_t^+ is the best point observed so far.

In most cases, acquisition functions provide knobs (e.g., κ) for controlling the exploration-exploitation trade-off.

- Search in regions where $\mu_{GP}(x)$ is high (exploitation)
- Probe regions where uncertainty $\sigma_{GP}(x)$ is high (exploration)

... and repeat until convergence ($t = 2$)



BAYES OPT IN A NUTSHELL

[slides from Gilles Louppe]

Bayesian optimisation

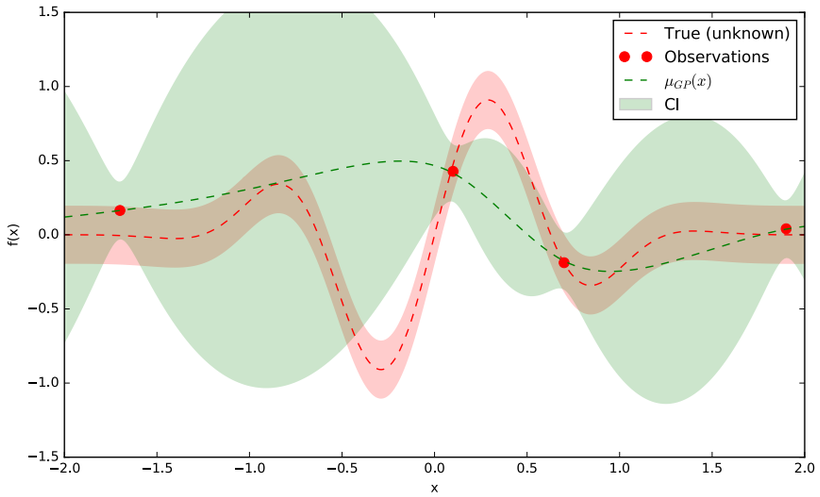
for $t = 1 : T$,

1. Given observations (x_i, y_i) for $i = 1 : t$, build a probabilistic model for the objective f.
   - Integrate out all possible true functions, using Gaussian process regression.
2. Optimise a cheap utility function u based on the posterior distribution for sampling the next point.

$$x_{t+1} = \arg\max_x u(x)$$

   Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

Build a probabilistic model for the objective function



This gives a posterior distribution over functions that could have generated the observed data.

Acquisition functions

Acquisition functions u(x) specify which sample x should be tried next:

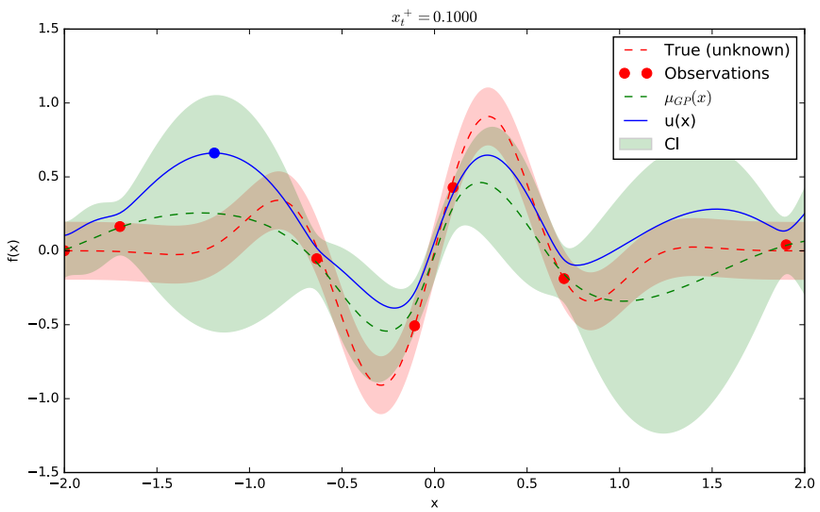
- Upper confidence bound $\text{UCB}(x) = \mu_{GP}(x) + \kappa\sigma_{GP}(x)$;
- Probability of improvement $\text{PI}(x) = P(f(x) \geq f(x_t^+) + \kappa)$;
- Expected improvement $\text{EI}(x) = \mathbb{E}[f(x) - f(x_t^+)]$;
- ... and many others.

where x_t^+ is the best point observed so far.

In most cases, acquisition functions provide knobs (e.g., κ) for controlling the exploration-exploitation trade-off.

- Search in regions where $\mu_{GP}(x)$ is high (exploitation)
- Probe regions where uncertainty $\sigma_{GP}(x)$ is high (exploration)

... and repeat until convergence ($t = 3$)



BAYES OPT IN A NUTSHELL

[slides from Gilles Louppe]

Bayesian optimisation

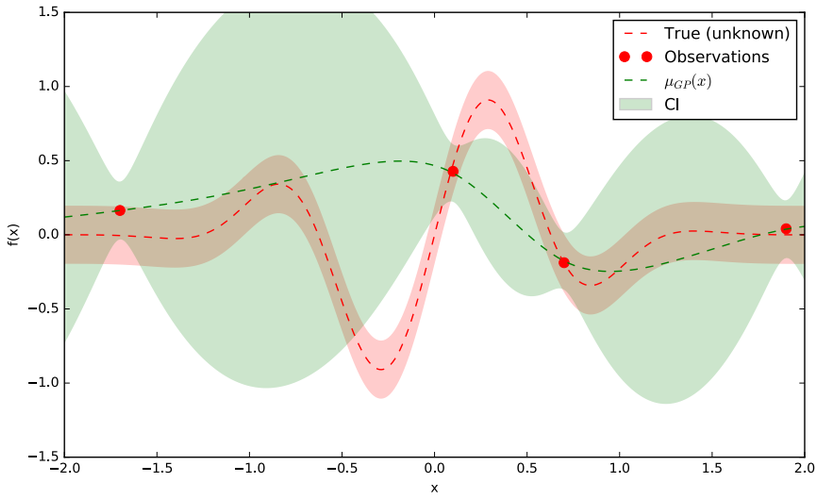
for $t = 1 : T$,

1. Given observations (x_i, y_i) for $i = 1 : t$, build a probabilistic model for the objective f.
   - Integrate out all possible true functions, using Gaussian process regression.
2. Optimise a cheap utility function u based on the posterior distribution for sampling the next point.

$$x_{t+1} = \arg\max_x u(x)$$

   Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

Build a probabilistic model for the objective function



This gives a posterior distribution over functions that could have generated the observed data.

Acquisition functions

Acquisition functions u(x) specify which sample x should be tried next:

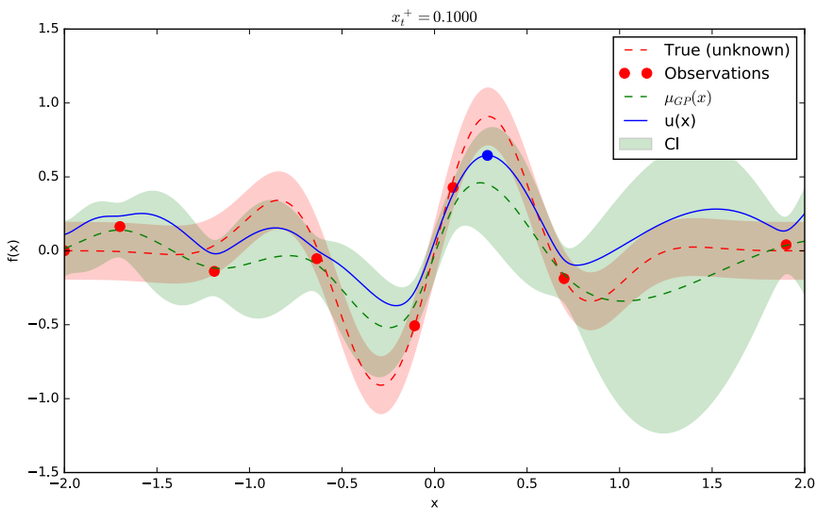
- Upper confidence bound $\text{UCB}(x) = \mu_{GP}(x) + \kappa\sigma_{GP}(x)$;
- Probability of improvement $\text{PI}(x) = P(f(x) \geq f(x_t^+) + \kappa)$;
- Expected improvement $\text{EI}(x) = \mathbb{E}[f(x) - f(x_t^+)]$;
- ... and many others.

where x_t^+ is the best point observed so far.

In most cases, acquisition functions provide knobs (e.g., κ) for controlling the exploration-exploitation trade-off.

- Search in regions where $\mu_{GP}(x)$ is high (exploitation)
- Probe regions where uncertainty $\sigma_{GP}(x)$ is high (exploration)

... and repeat until convergence ($t = 4$)



BAYES OPT IN A NUTSHELL

[slides from Gilles Louppe]

Bayesian optimisation

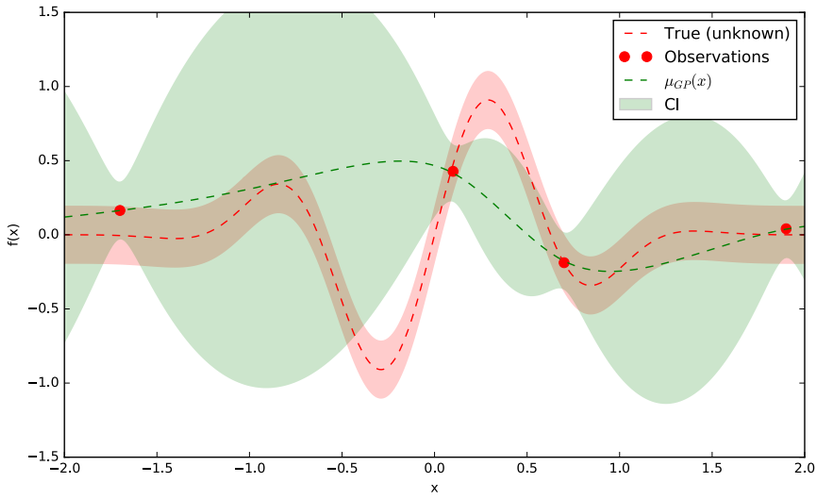
for $t = 1 : T$,

1. Given observations (x_i, y_i) for $i = 1 : t$, build a probabilistic model for the objective f.
   - Integrate out all possible true functions, using Gaussian process regression.
2. Optimise a cheap utility function u based on the posterior distribution for sampling the next point.

$$x_{t+1} = \arg\max_x u(x)$$

   Exploit uncertainty to balance exploration against exploitation.
3. Sample the next observation y_{t+1} at x_{t+1}.

Build a probabilistic model for the objective function



This gives a posterior distribution over functions that could have generated the observed data.

Acquisition functions

Acquisition functions u(x) specify which sample x should be tried next:

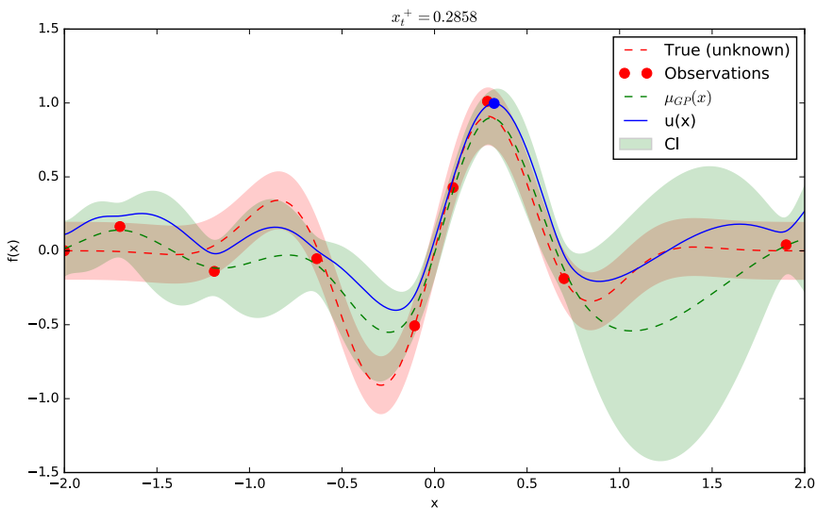
- Upper confidence bound $\text{UCB}(x) = \mu_{GP}(x) + \kappa\sigma_{GP}(x)$;
- Probability of improvement $\text{PI}(x) = P(f(x) \geq f(x_t^+) + \kappa)$;
- Expected improvement $\text{EI}(x) = \mathbb{E}[f(x) - f(x_t^+)]$;
- ... and many others.

where x_t^+ is the best point observed so far.

In most cases, acquisition functions provide knobs (e.g., κ) for controlling the exploration-exploitation trade-off.

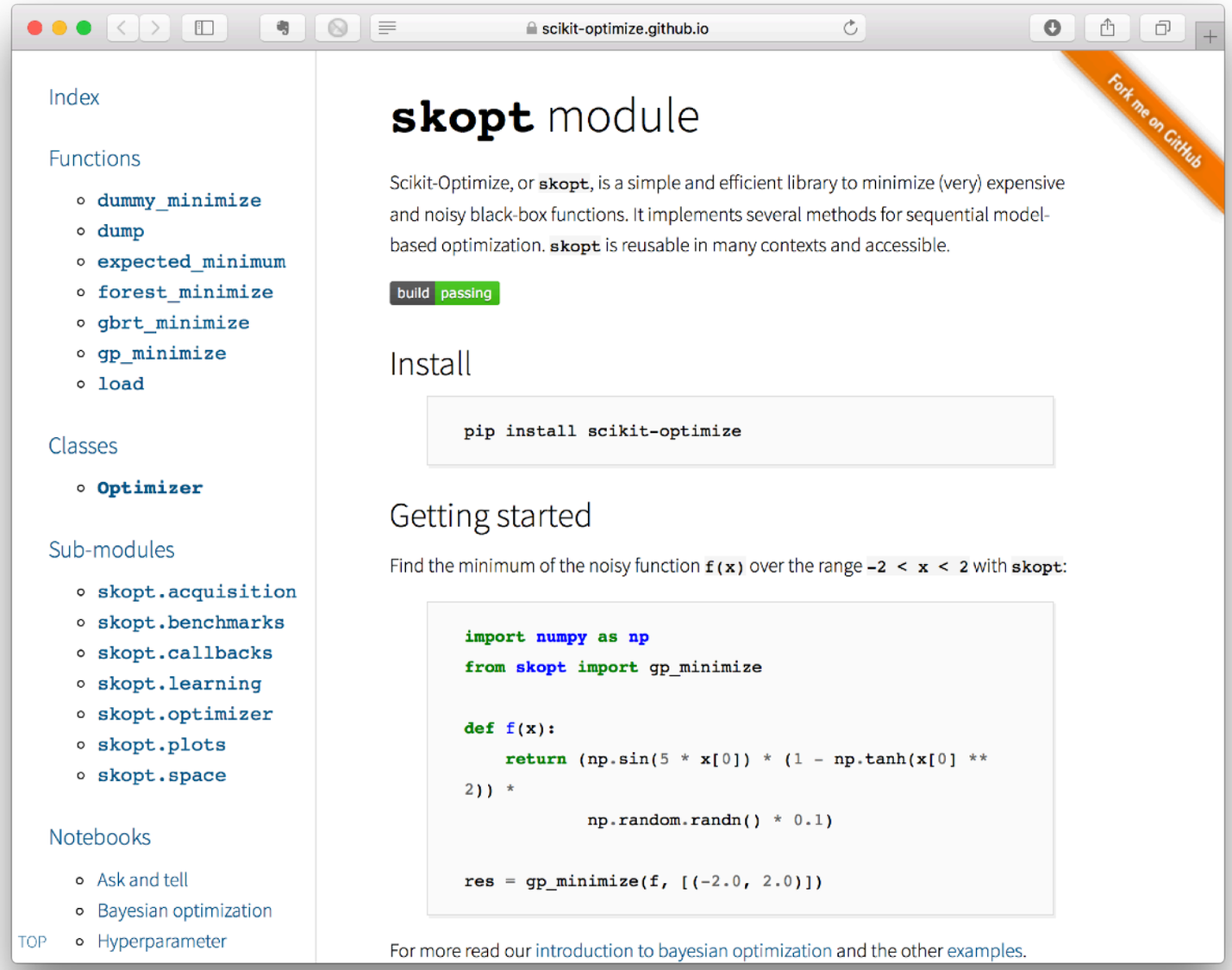
- Search in regions where $\mu_{GP}(x)$ is high (exploitation)
- Probe regions where uncertainty $\sigma_{GP}(x)$ is high (exploration)

... and repeat until convergence ($t = 5$)



OPTIMIZATION SOFTWARE

- Python
  - Spearmint https://github.com/JasperSnoek/spearmint
  - GPyOpt https://github.com/SheffieldML/GPyOpt
  - RoBO https://github.com/automl/RoBO
  - scikit-optimize https://github.com/MechCoder/scikit-optimize (work in progress)
- C++
  - MOE https://github.com/yelp/MOE



GitHub Repo for previous slides:
https://github.com/glouppe/talk-bayesian-optimisation

SOFTWARE

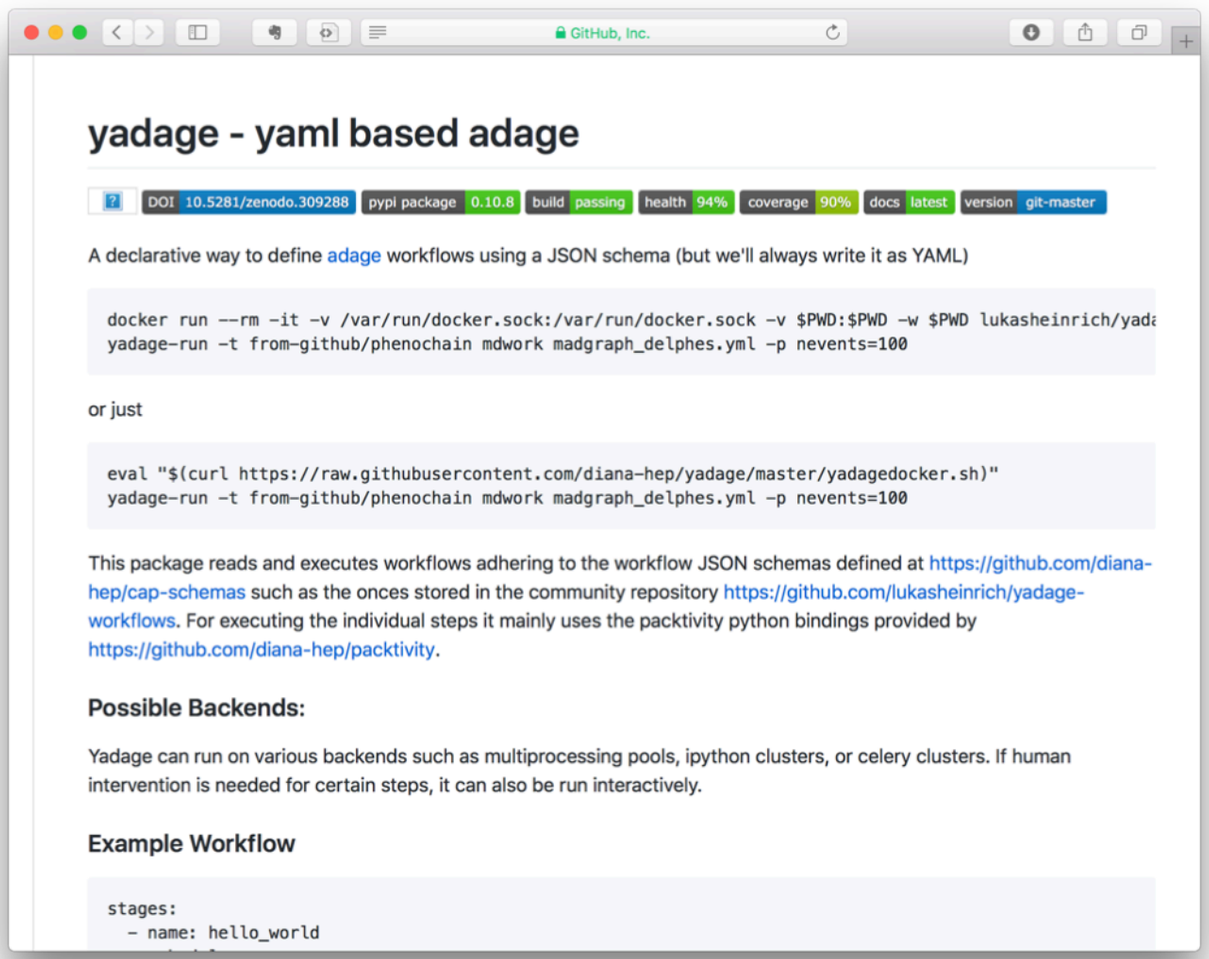
Yadage and Packtivity – analysis preservation using parametrized workflows

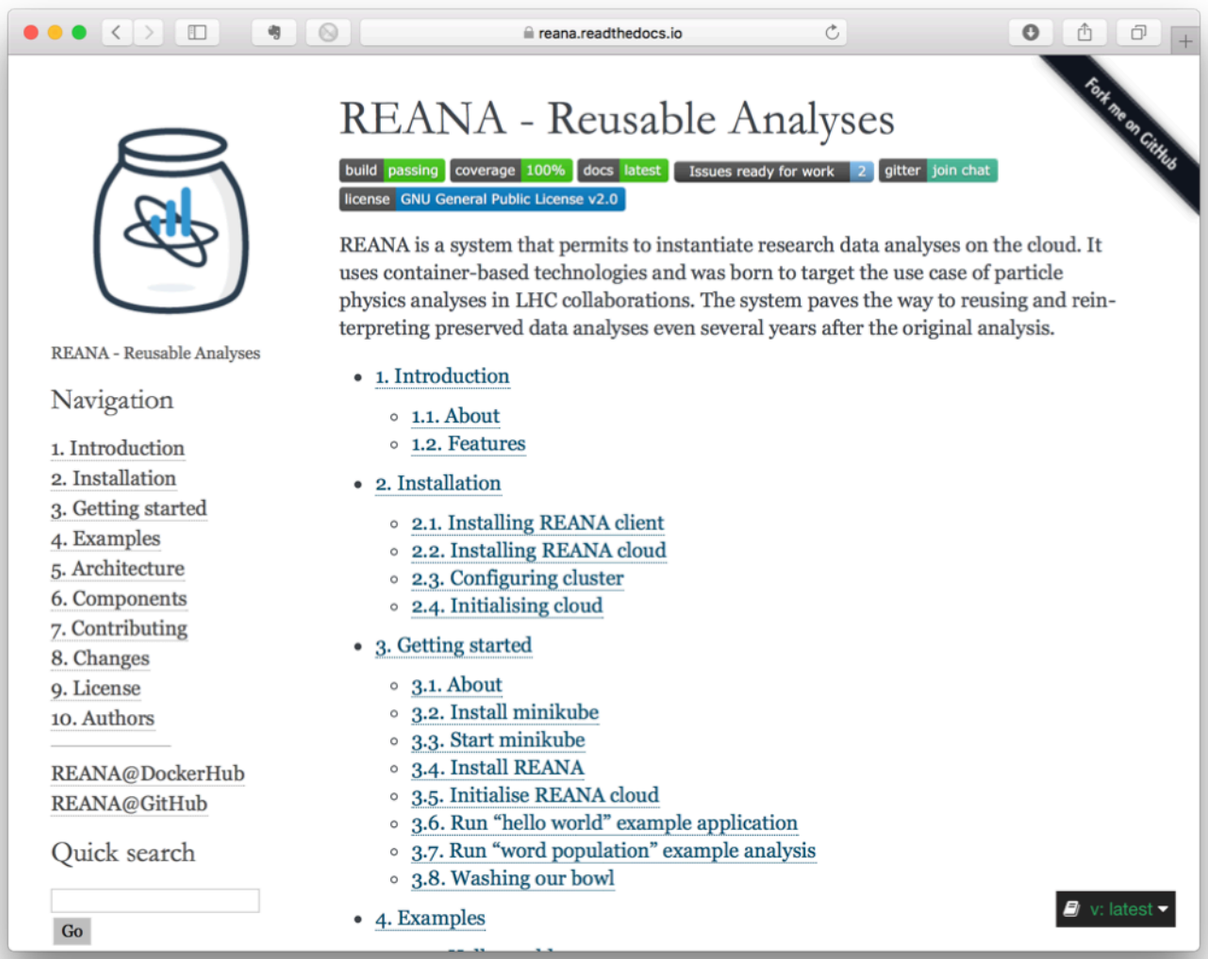
Kyle Cranmer[1] **and Lukas Heinrich**[1]

[1] Department of Physics, New York University, New York, USA

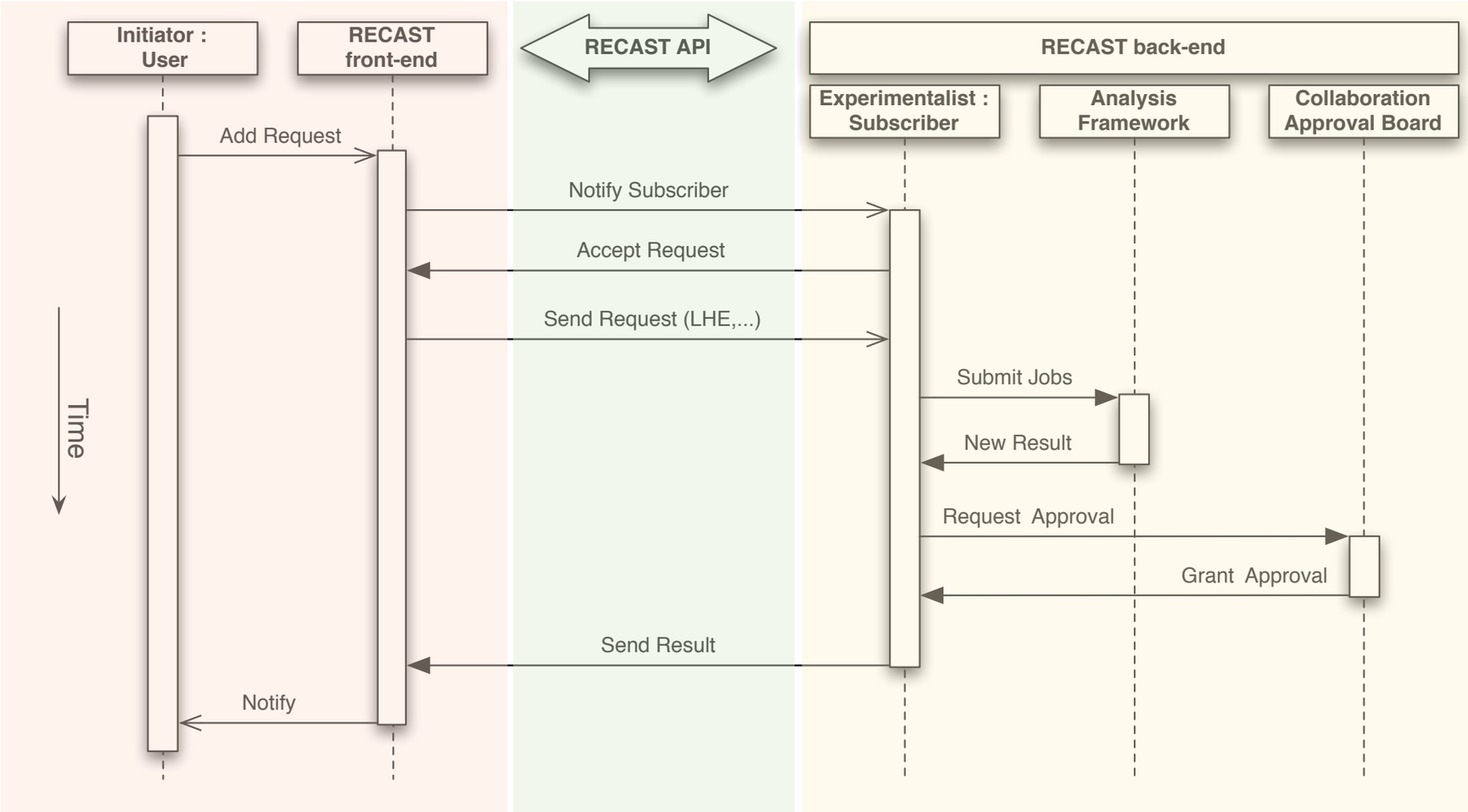
E-mail: `lukas.heinrich@cern.ch`

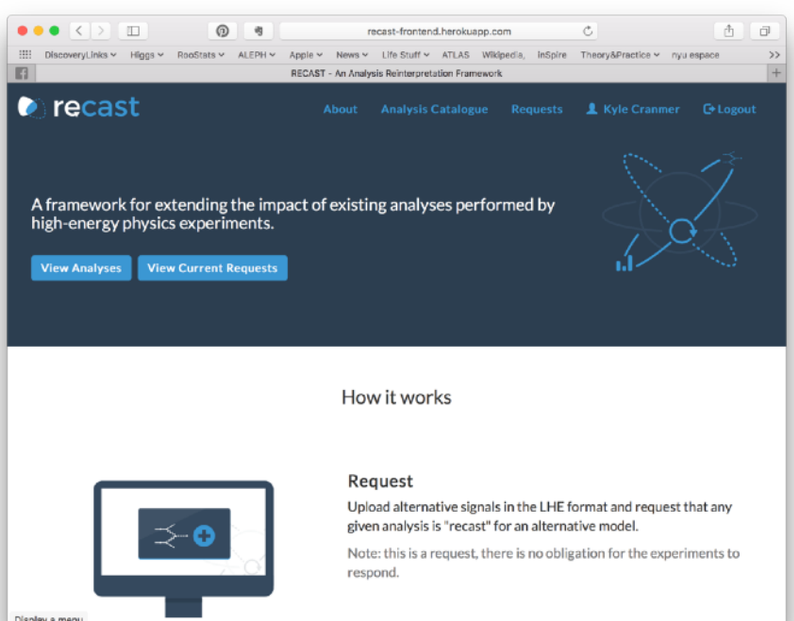
Abstract. Preserving data analyses produced by the collaborations at LHC in a parametrized fashion is crucial in order to maintain reproducibility and re-usability. We argue for a declarative description in terms of individual processing steps – "packtivities" – linked through a dynamic directed acyclic graph (DAG) and present an initial set of JSON schemas for such a description and an implementation – "yadage" – capable of executing workflows of analysis preserved via Linux containers.



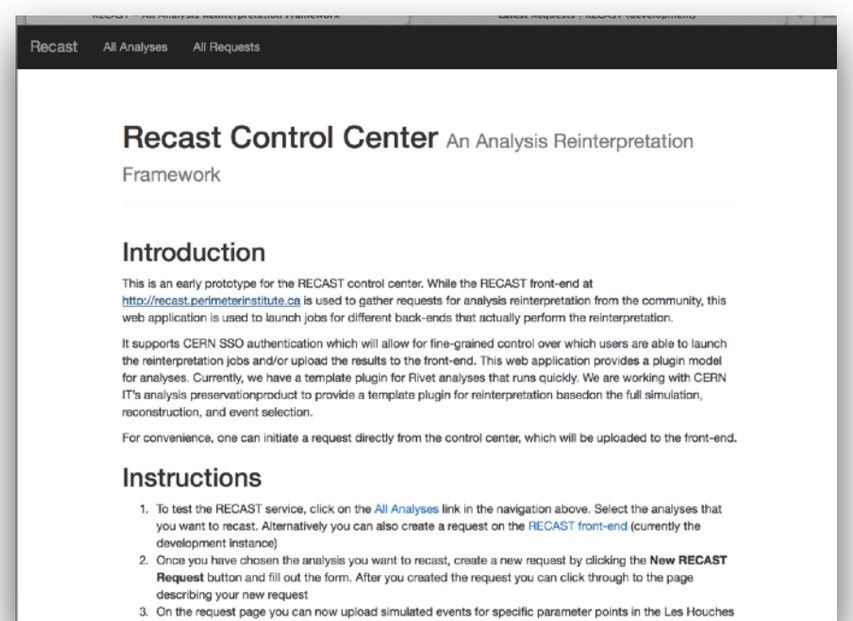


https://arxiv.org/pdf/1706.01878.pdf

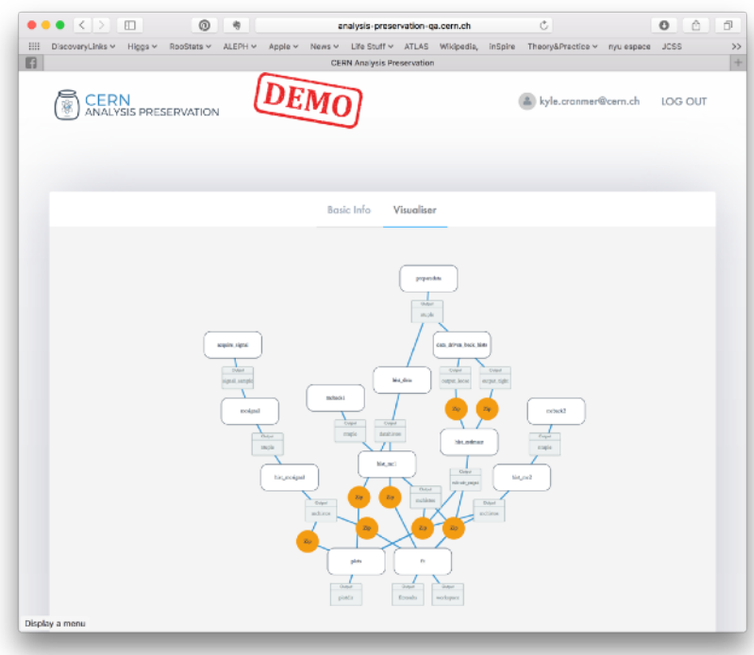




Front-End: public facing
collects requests



Control Center: not public, uses CERN auth.,
oversees processing of jobs on back-end



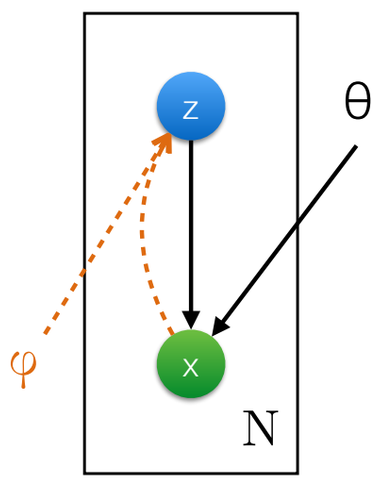
CERN Analysis Preservation:
Stores workflows, provides back-end
computing resources

VARIATIONAL AUTO-ENCODER

[Slides from D. Kingma NIPS 2015]

Auto-Encoding Variational Bayes

[Kingma and Welling, 2013/2014]
[Rezende et al, 2014]



- $q_\varphi(z|x) = N(\mu,\sigma^2)$
  $[\mu, \sigma^2] = f^{(z|x)}(x,\varphi) = \text{multilayer neural net}$
- Objective: lower bound of log p(x).
  - Jointly optimized w.r.t. φ and θ
  - This is approx. maximum likelihood
  - Simple SGD:
    - Sampling small minibatches of data
    - Sampling from approx. posterior
- This also minimizes an expected KL divergence
  $D_{KL}(q_\varphi(z|x)||p(z|x))$
  -> gives us cheap approx. inference for new datapoints

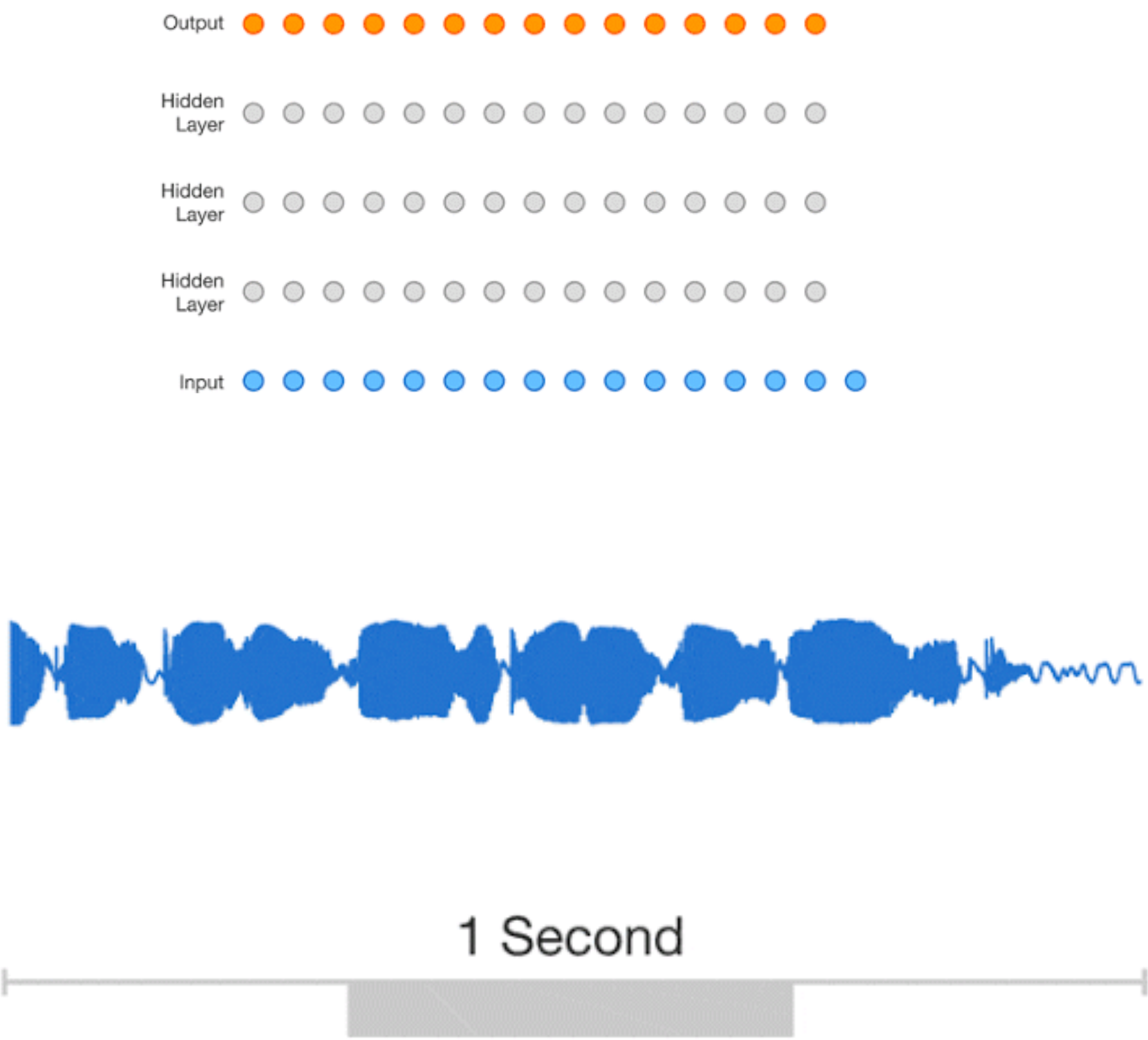
Diederik (Durk) Kingma

Max Welling

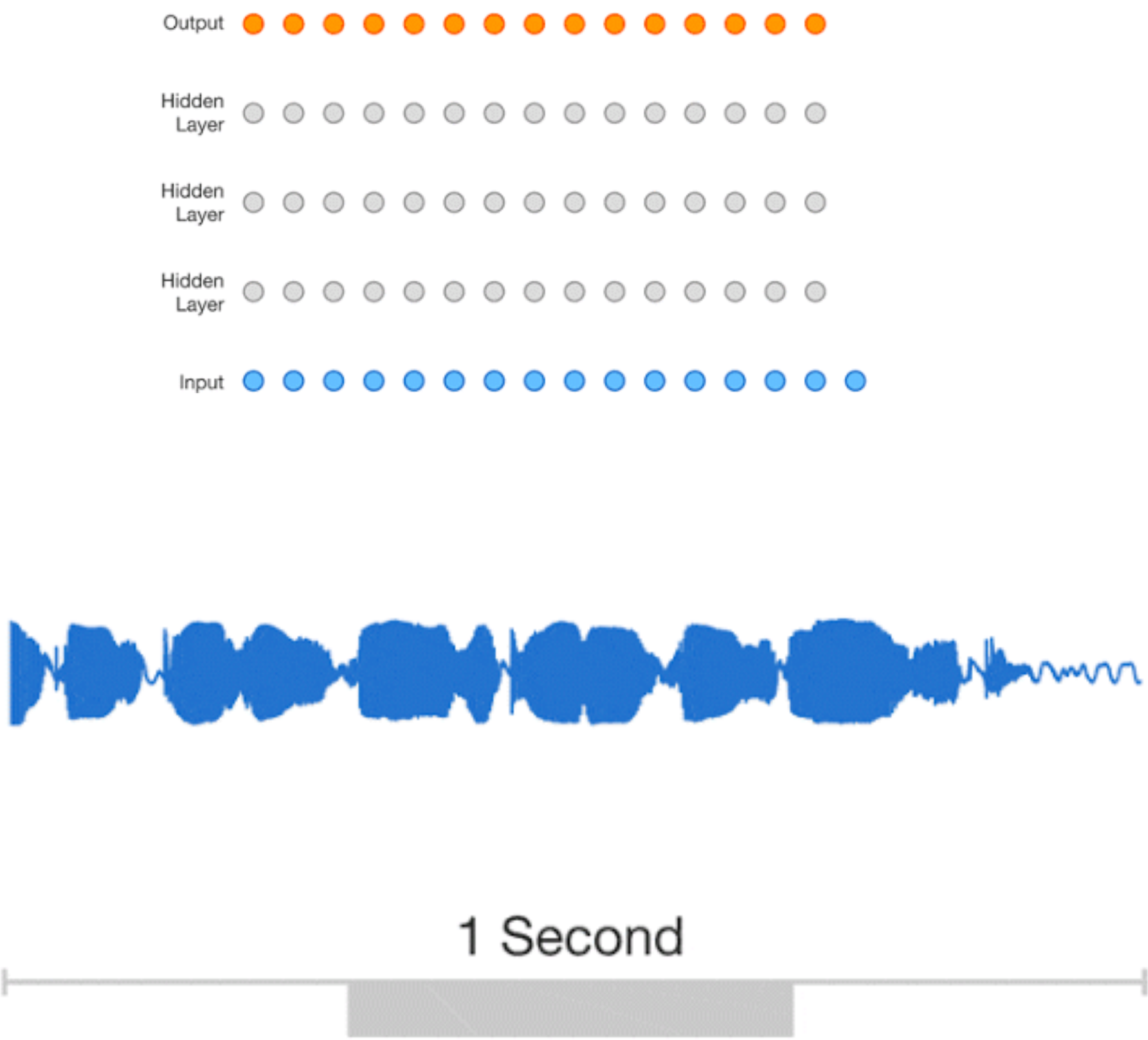
Conv. net as encoder/decoder, trained on faces

Kingma and Welling, Auto-encoding Variational Bayes, ICLR 2014
Rezende, Mohamed and Wierstra, Stochastic back-propagation and variational inference in deep latent Gaussian models, ICML 2014

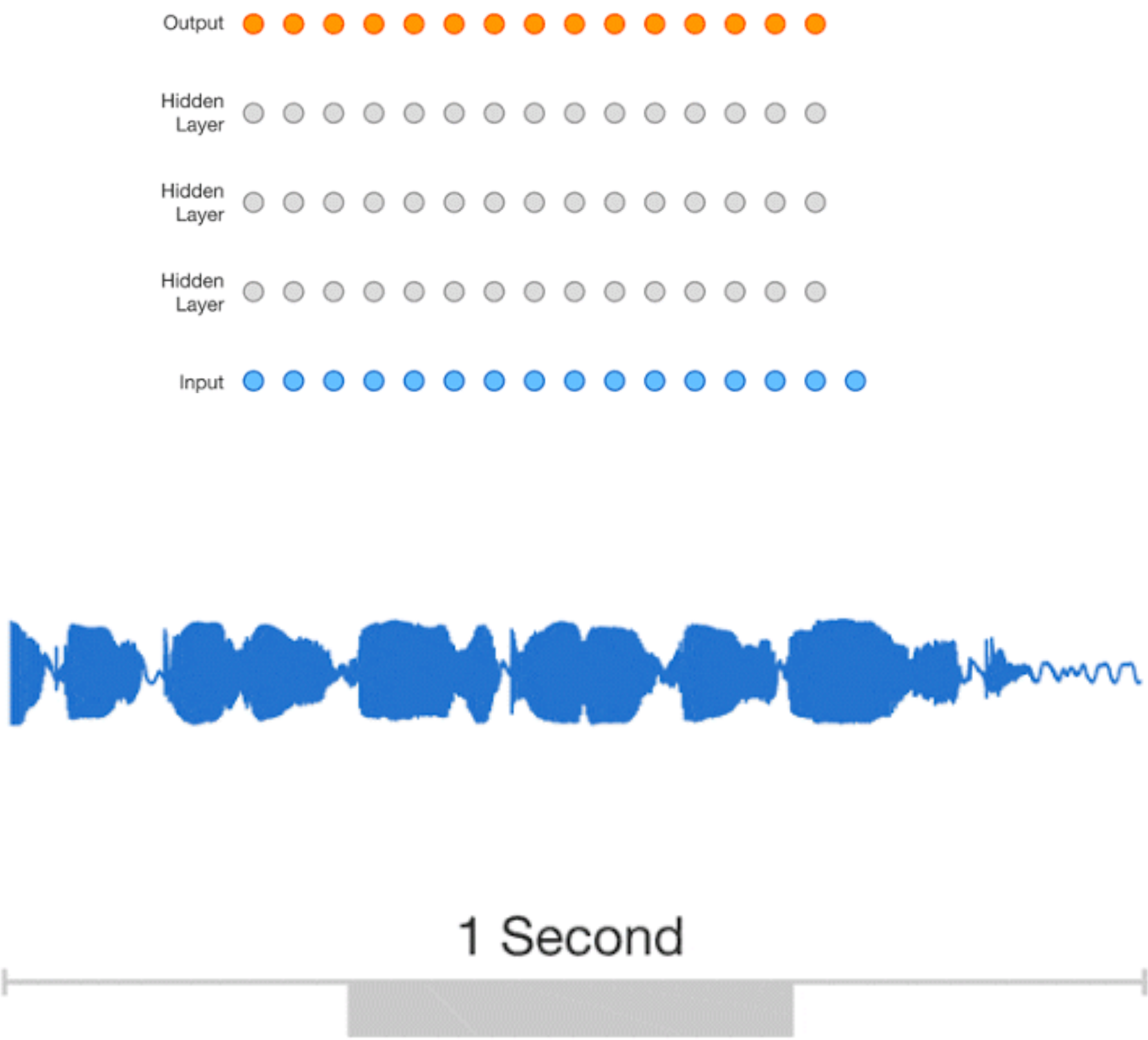
WAVENET: A GENERATIVE MODEL FOR RAW AUDIO



WAVENET: A GENERATIVE MODEL FOR RAW AUDIO



WAVENET: A GENERATIVE MODEL FOR RAW AUDIO



Create standalone simulation tools to facilitate collaboration between HEP and machine learning community

By Pierre Baldi, Peter Sadowski, Daniel Whiteson, Christian Lorenz Müller, Michael Williams, Lukas Heinrich, Steven Schramm, Maurizio Pierini, Sergei Gleyzer, Amir Farbin, jean-roch vlimant, Tim Head, Juan Pavez, Peter Elmer, Balázs Kégl, Andrey Ustyuzhanin, Vladimir Gligorov, Gilles Louppe, Kyle Cranmer

dslhc | machinelearning | datascience | open data | simulation

Discussions at recent workshops have made it clear that one of the key barriers to collaboration between high energy physics and the machine learning community is access to training data. Recent successes in data sharing through the HiggsML and Flavours of Physics Kaggle challenges have borne much fruit, but required significant effort to coordinate.

While static simulated datasets are useful for challenges, in the course of investigating new machine learning techniques it is advantageous to be able to generate training data on demand (e.g. Refs. 1, 2, 3).
Therefore we recommend efforts be made to produce the ingredients required to facilitate such collaboration:

- Specific challenges for HEP experiments should be fully specified such that minimal domain-specific knowledge is required to attack them.
- Stand-alone simulators should be made open source. They should be developed to be easy to use without domain-specific expertise, while still being representative of real experimental challenges. Such a simulation will permit non-HEP researchers to generate realistic HEP datasets for training and testing. These simulators could range from truth-level simulation of a hard scattering to fast simulation like Delphes, to full GEANT4 simulation of sensor arrays.
- Performance metrics (objective functions) and operational constraints should be defined to evaluate proposed solutions.

Kyle Cranmer · Sign out

Actions

1 vote | Hide | Collect

Share 29 | Tweet 9

Authors

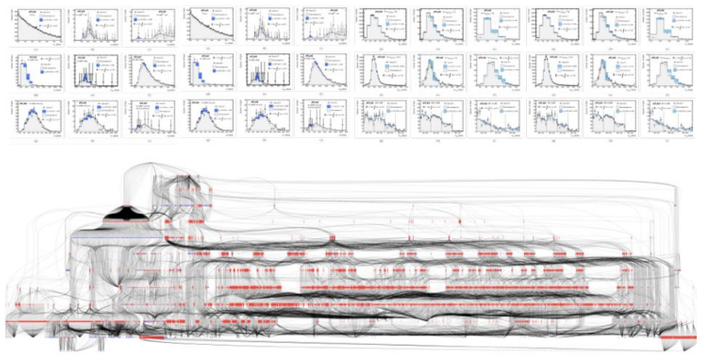
Pierre Baldi, Peter Sadowski, Daniel Whiteson, Christian Lorenz Müller, Michael Williams, Lukas Heinrich, Steven Schramm, Maurizio Pierini, Sergei Gleyzer, Amir Farbin, jean-roch vlimant, Tim Head, Juan Pavez, Peter Elmer, Balázs Kégl, Andrey Ustyuzhanin, Vladimir Gligorov, Gilles Louppe, Kyle Cranmer

Metadata

DOI 10.5281/zenodo.46864

Published: 26 Feb, 2016

Probabilistic programming frameworks

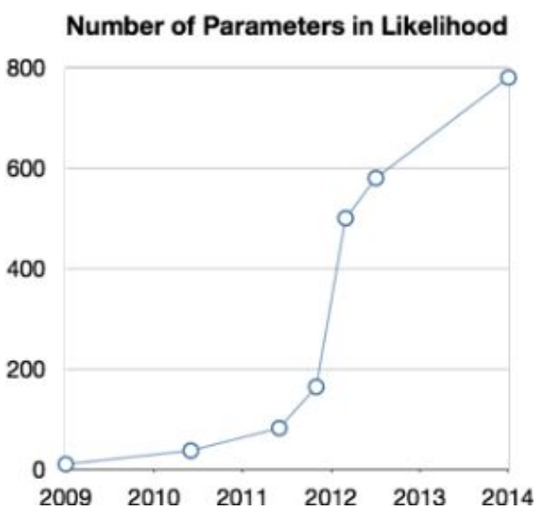


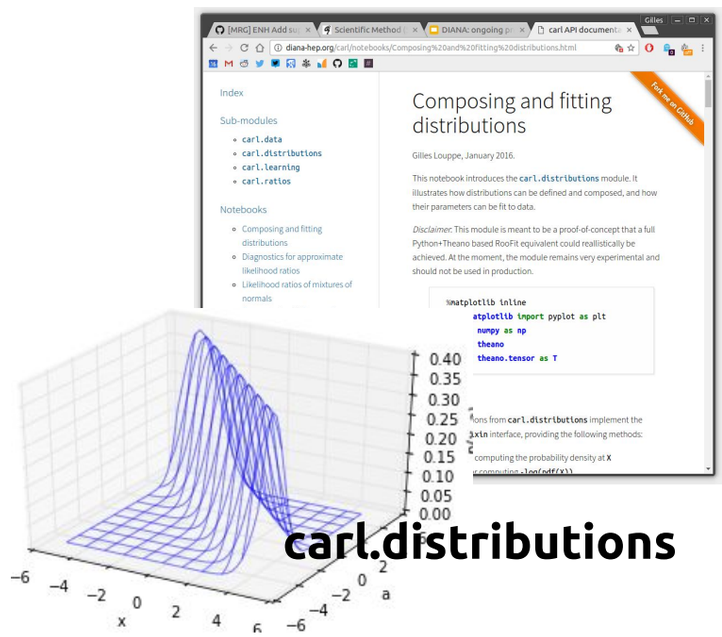
$$\mathbf{f}_{\text{tot}}(\mathcal{D}_{\text{sim}}, \mathcal{G}|\boldsymbol{\alpha}) = \prod_{c \in \text{channels}} \left[\text{Pois}(n_c|\nu_c(\boldsymbol{\alpha})) \prod_{e=1}^{n_c} f_c(x_{ce}|\boldsymbol{\alpha}) \right] \cdot \prod_{p \in \mathbb{S}} f_p(a_p|\alpha_p)$$

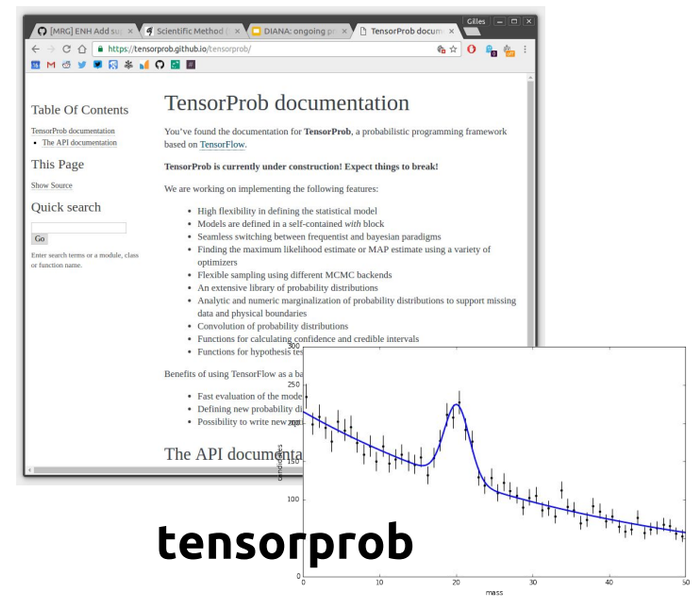
RooFit serves us well, but shows limits in terms of **scalability**.

Using a data flow graph framework, RooFit would be **distributed**, **GPU-enabled** and automatically **differentiable**.

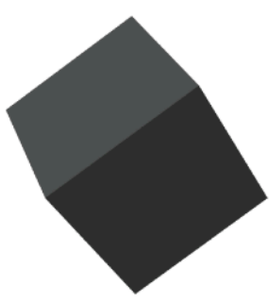
Feasibility? Certainly **within reach!** As illustrated by our tentative proof-of-concepts carl.distributions [Gilles Louppe] and tensorprob [Igor Babuschkin, now at DeepMind]. See also Edward.







Edward



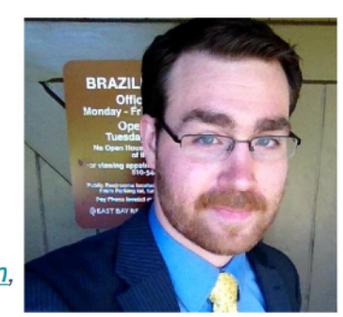
A library for probabilistic modeling, inference, and criticism.

Edward is a Python library for probabilistic modeling, inference, and criticism. It is a testbed for fast experimentation and research with probabilistic models, ranging from classical hierarchical models on small data sets to complex deep probabilistic models on large data sets. Edward fuses three fields: Bayesian statistics and machine learning, deep learning, and probabilistic programming.

It supports **modeling** with

Dustin Tran

Ph.D. Student
Columbia University
dustin@cs.columbia.edu (@dustinvtran, http://dustintran.com



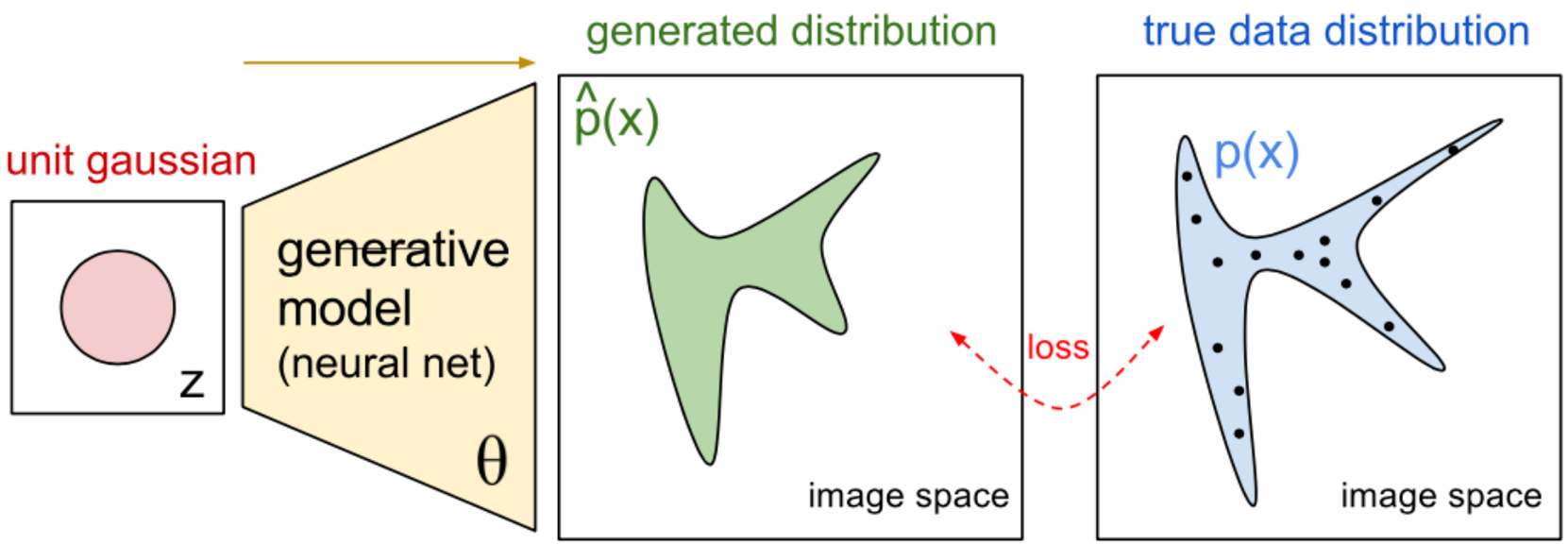
Matthew Feickert

High Energy Physics Ph.D. Candidate
Southern Methodist University
matthew.feickert@cern.ch or mfeickert@smu.edu
GitHub: matthewfeickert @HEPfeickert

Adversarial Training
(not just for GANs)

GENERATIVE ADVERSARIAL NETWORKS

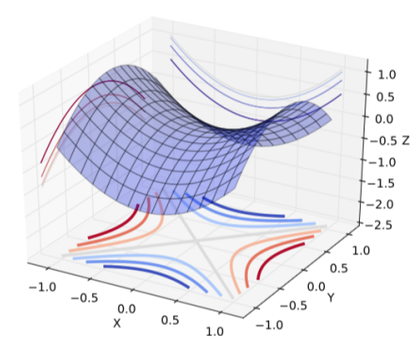
Goodfellow, et al arXiv:1406.2661
slide Gilles Louppe



Leo is G Tom is D

- Two-player game:
  - a discriminator D,
  - a generator G;
- D is a classifier $\mathcal{X} \mapsto \{0, 1\}$ that tries to distinguish between
  - a sample from the data distribution ($D(\mathbf{x}) = 1$, for $\mathbf{x} \sim p_{\text{data}}$),
  - and a sample from the model distribution ($D(G(\mathbf{z})) = 0$, for $\mathbf{z} \sim p_{\text{noise}}$);
- G is a generator $\mathcal{Z} \mapsto \mathcal{X}$ trained to produce samples $G(\mathbf{z})$ (for $\mathbf{z} \sim p_{\text{noise}}$) that are difficult for D to distinguish from data.

$$(D^*, G^*) = \max_D \min_G V(D, G).$$



NEW! AVO

G. Louppe & K.C. arXiv:1707.07113

Adversarial Variational Optimization of Non-Differentiable Simulators

Gilles Louppe[1] and **Kyle Cranmer**[1]
[1]*New York University*

Complex computer simulators are increasingly used across fields of science as generative models tying parameters of an underlying theory to experimental observations. Inference in this setup is often difficult, as simulators rarely admit a tractable density or likelihood function. We introduce Adversarial Variational Optimization (AVO), a likelihood-free inference algorithm for fitting a non-differentiable generative model incorporating ideas from empirical Bayes and variational inference. We adapt the training procedure of generative adversarial networks by replacing the differentiable generative network with a domain-specific simulator. We solve the resulting non-differentiable minimax problem by minimizing variational upper bounds of the two adversarial objectives. Effectively, the procedure results in learning a proposal distribution over simulator parameters, such that the corresponding marginal distribution of the generated data matches the observations. We present results of the method with simulators producing both discrete and continuous data.

Leo is *G* Tom is *D*

Similar to GAN setup, but instead of using a neural network as the generator, use the actual simulation (eg. Pythia, GEANT)

Continue to use a neural network discriminator / critic.

Difficulty: the simulator isn't differentiable, but there's a **trick**!

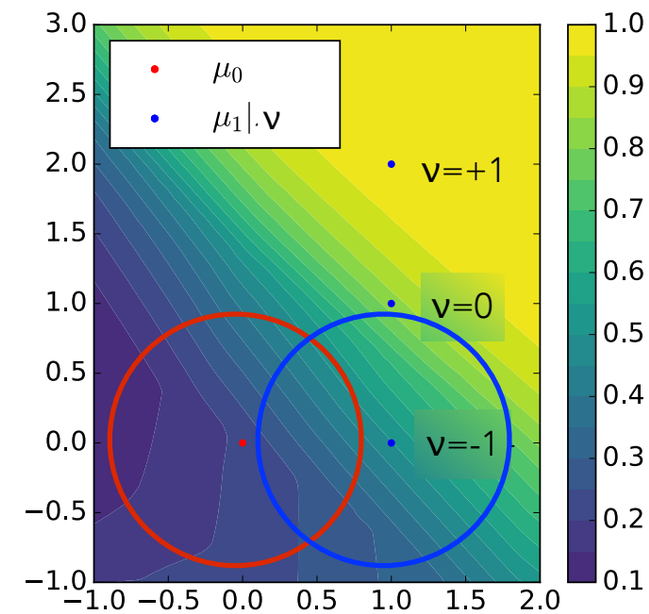
Allows us to efficiently fit / **tune simulation** with stochastic gradient techniques!

LEARNING TO PIVOT WITH ADVERSARIAL NETWORKS

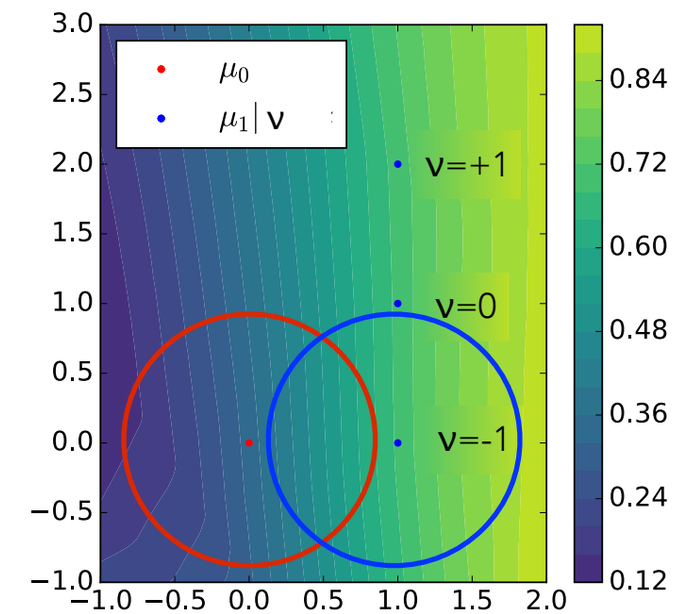
Typically classifier **f(x)** trained to minimize loss $\mathbf{L_f}$.

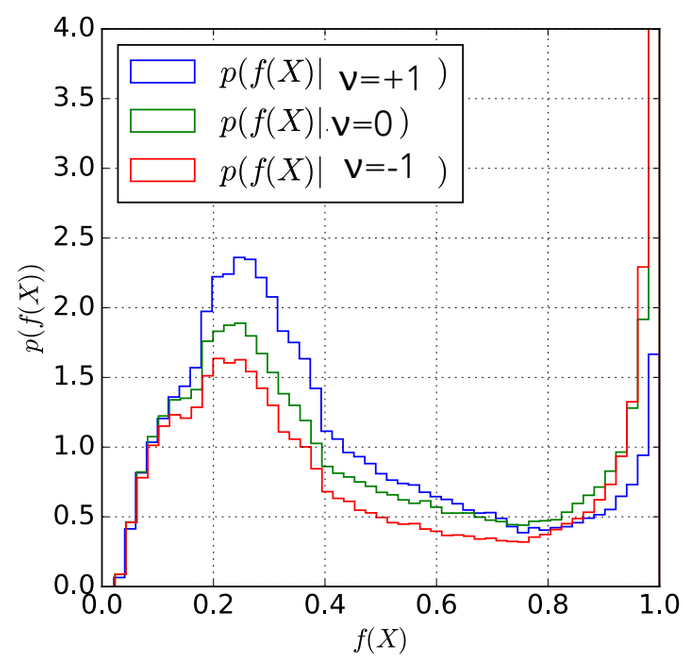
- want classifier output to be insensitive to systematics (nuisance parameter **ν**)
- introduce an **adversary r** that tries to predict ν based on f.
- provides training procedure that allows for **tradeoff** between traditional classification accuracy and **robustness to systematics**

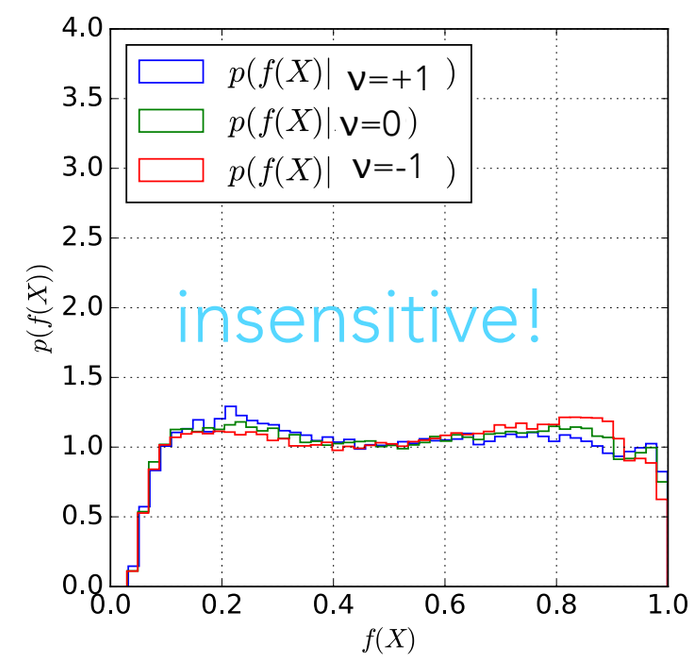
normal training



adversarial training







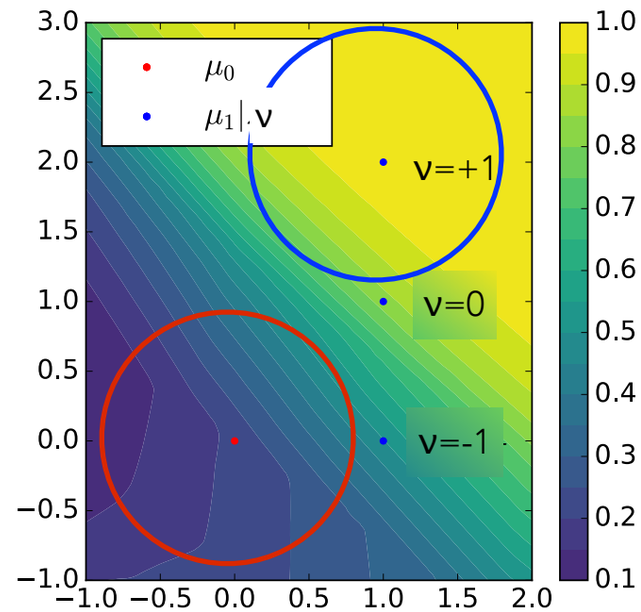
G. Louppe, M. Kagan, K. Cranmer, arXiv:1611.01046

LEARNING TO PIVOT WITH ADVERSARIAL NETWORKS

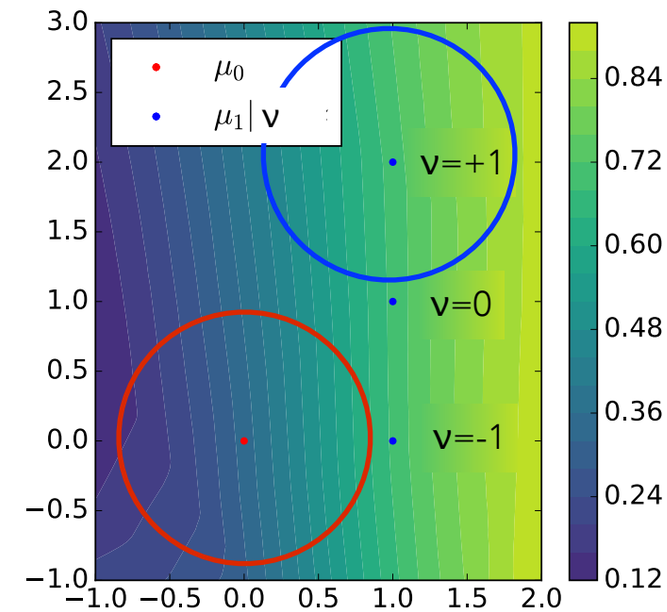
Typically classifier **f(x)** trained to minimize loss $\mathbf{L_f}$.

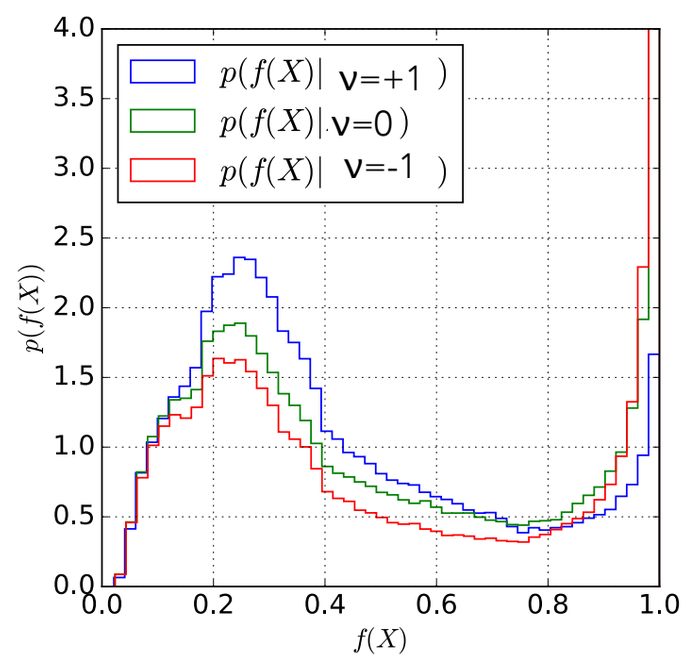
- want classifier output to be insensitive to systematics (nuisance parameter **ν**)
- introduce an **adversary r** that tries to predict ν based on f.
- provides training procedure that allows for **tradeoff** between traditional classification accuracy and **robustness to systematics**

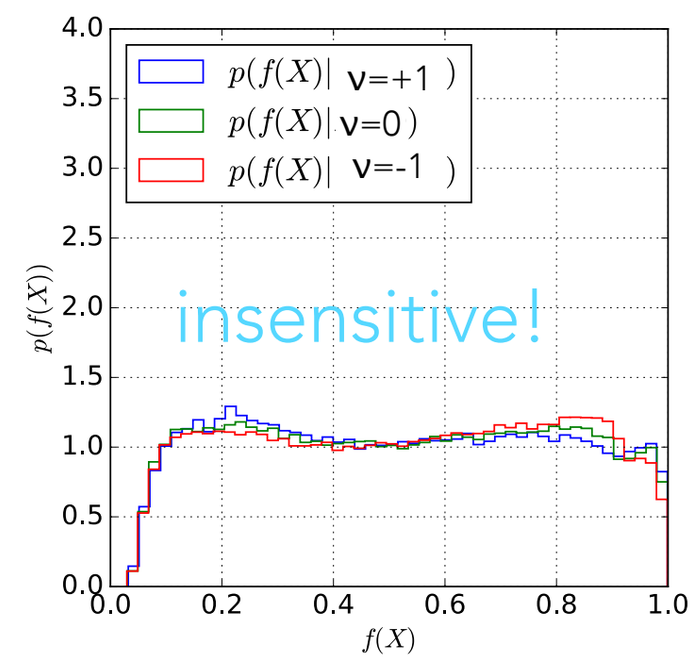
normal training



adversarial training







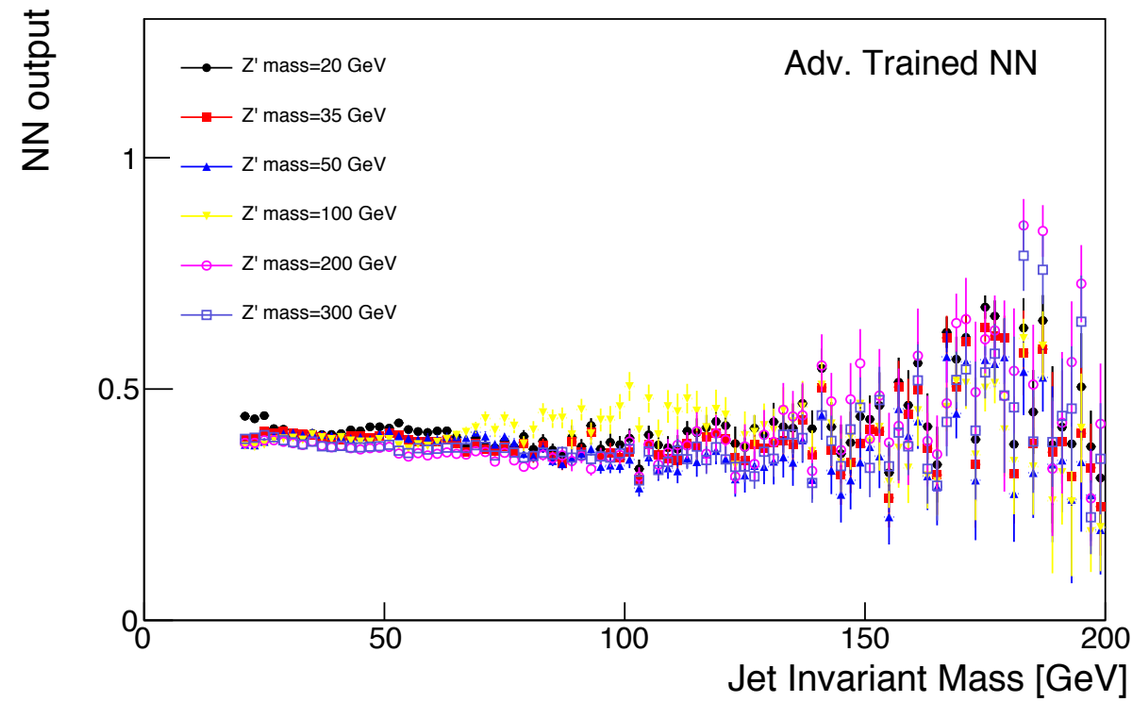
G. Louppe, M. Kagan, K. Cranmer, arXiv:1611.01046

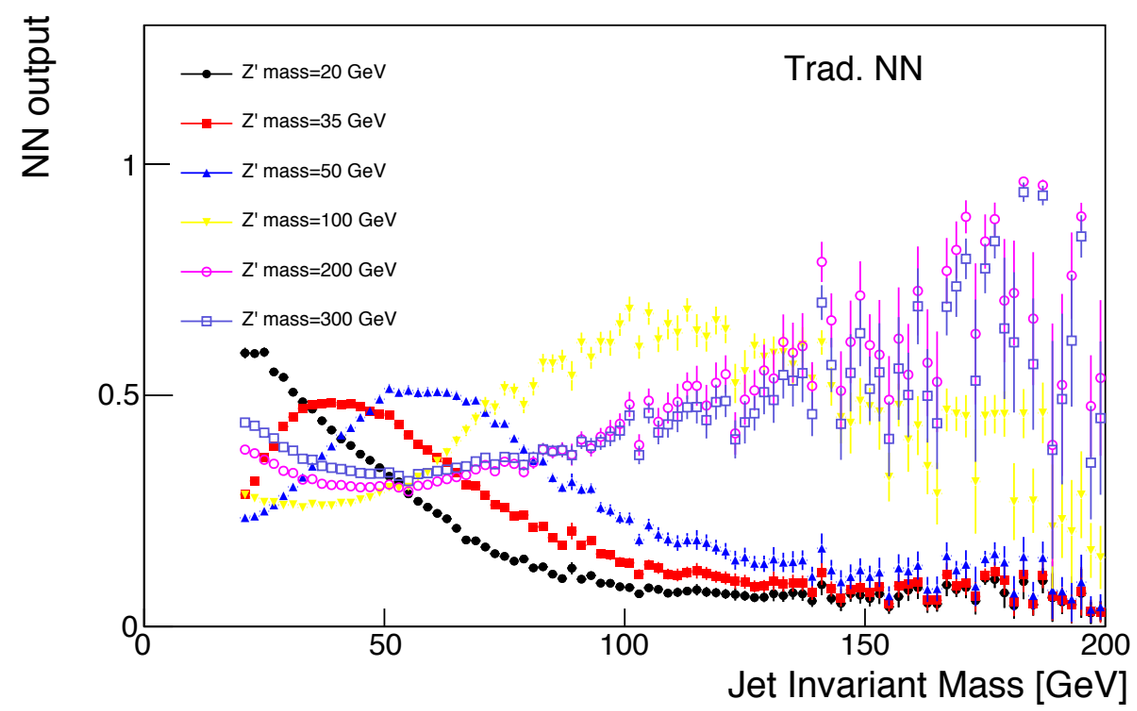
DECORRELATED TAGGERS

K.C, J. Pavez, and G. Louppe, arXiv:1506.02169
P. Baldi, K.C, T. Faucett, P. Sadowski, D. Whiteson arXiv:1601.07913
G. Louppe, M. Kagan, K.C, arXiv:1611.01046
Shimmin, et. al. arXiv:1703.03507

Adversarial approach of "Learning to Pivot" can also be used to train a classifier that is "decorrelated" to some other variable.

- want jet taggers that are decorrelated with jet invariant mass
- so that analysis can still search for a bump using jet invariant mass
- avoids sculpting background





From Reproducibility
To Reusability

[work with Lukas Heinrich]

REINTERPRETATION

The BSM-AI project: SUSY-AI – generalizing LHC limits on supersymmetry with machine learning

Sascha Caron,[a,b] **Jong Soo Kim,**[c] **Krzysztof Rolbiecki,**[c,d]
Roberto Ruiz de Austri,[e] **Bob Stienen**[a]

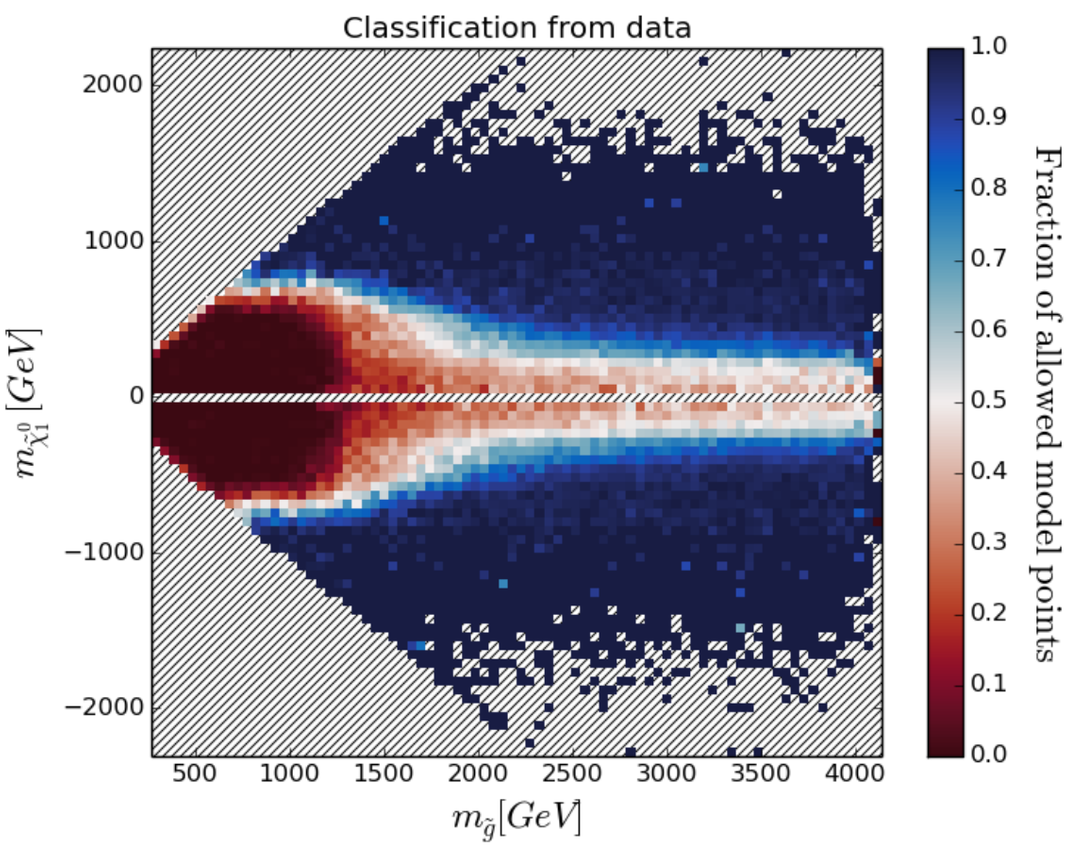
[a] *Institute for Mathematics, Astro- and Particle Physics IMAPP, Radboud Universiteit, Nijmegen, The Netherlands*

[b] *Nikhef, Amsterdam, The Netherlands*

[c] *Instituto de Física Teórica UAM/CSIC, Madrid, Spain*

[d] *Faculty of Physics, University of Warsaw, Warsaw, Poland*

[e] *Instituto de Física Corpuscular, IFIC-UV/CSIC, Valencia, Spain*



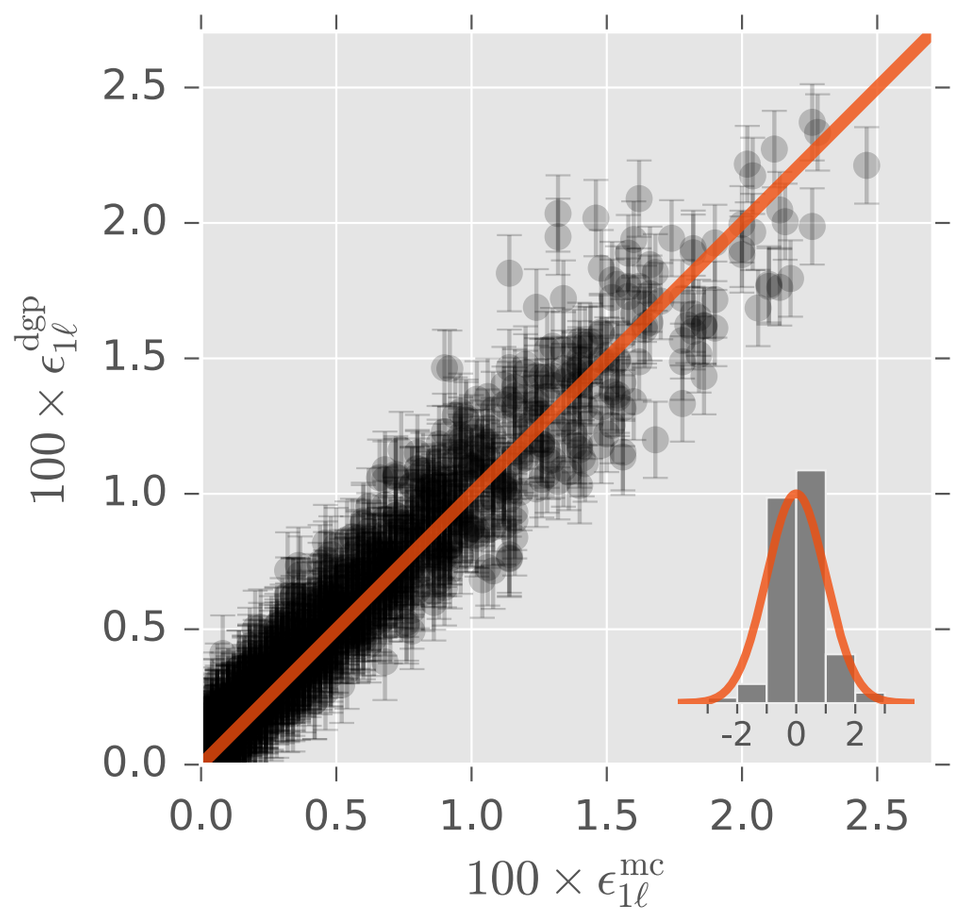
Accelerating the BSM interpretation of LHC data with machine learning

Gianfranco Bertone,[1] Marc Peter Deisenroth,[2] Jong Soo Kim,[3]
Sebastian Liem,[1] Roberto Ruiz de Austri,[4] and Max Welling[5]

[1] *GRAPPA, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, Netherlands*
[2] *Department of Computing, Imperial College London, 180 Queen's Gate, SW7 2AZ London, United Kingdom*
[3] *Center for Theoretical Physics of the Universe, Institute for Basic Science (IBS), Daejeon, 34051, Korea and Instituto de Física Teórica UAM/CSIC, Madrid, Spain*
[4] *Instituto de Física Corpuscular IFIC-UV/CSIC, Valencia, Spain*
[5] *Informatics Institute, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, Netherlands*

(Dated: November 10, 2016)

The interpretation of Large Hadron Collider (LHC) data in the framework of Beyond the Standard Model (BSM) theories is hampered by the need to run computationally expensive event generators and detector simulators. Performing statistically convergent scans of high-dimensional BSM theories is consequently challenging, and in practice unfeasible for very high-dimensional BSM theories. We present here a new machine learning method that accelerates the interpretation of LHC data, by learning the relationship between BSM theory parameters and data. As a proof-of-concept, we demonstrate that this technique accurately predicts natural SUSY signal events in two signal regions at the High Luminosity LHC, up to four orders of magnitude faster than standard techniques. The new approach makes it possible to rapidly and accurately reconstruct the theory parameters of complex BSM theories, should an excess in the data be discovered at the LHC.



It’s the difference between if you had airplanes where you threw away an airplane after every flight, versus you could reuse them multiple times.

— Elon Musk

analysis pipeline

analyses pipeline

testing

one theory

It's the difference between if you had ~~airplanes~~ ~~airplane~~ where you threw away an ~~airplane~~ after ~~every flight~~, versus you could reuse them multiple times.

— Elon Musk

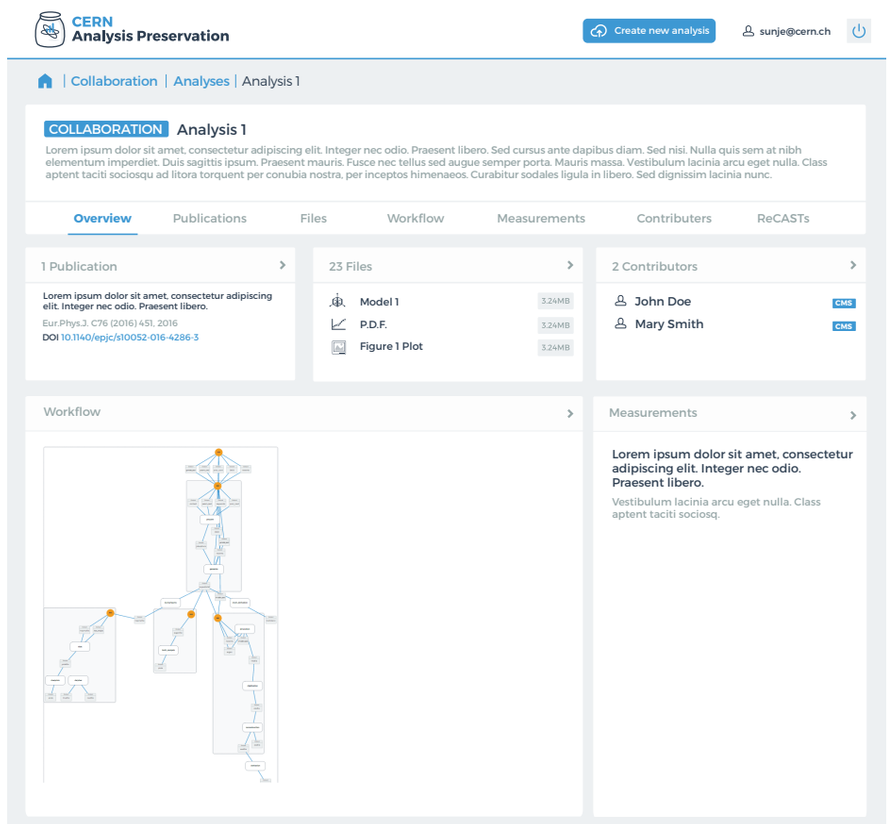
Technical Solution:

Workflow (i.e. logic which steps to run in which order: reconstruction → analysis→ fit)

- in easy to write / read text based format (YAML)
- generic workflow language “**yadage**” based on graphs. No assumption on how you run your analysis. Should be able to accommodate your workflows.
- integrated into CERN Analysis Preservation.
- re-run workflow using tool that interprets info stored in CAP

workflow	software	data
`eventsel.yml`	`docker img`	`data, bkgds`
`fit.yml`	`docker img`	
`workflow.yml`		

import analysis workflow



CERN Analysis Preservation

https://arxiv.org/pdf/1706.01878.pdf

SOFTWARE

Yadage and Packtivity – analysis preservation using parametrized workflows

Kyle Cranmer[1] **and Lukas Heinrich**[1]

[1] Department of Physics, New York University, New York, USA

E-mail: `lukas.heinrich@cern.ch`

Abstract. Preserving data analyses produced by the collaborations at LHC in a parametrized fashion is crucial in order to maintain reproducibility and re-usability. We argue for a declarative description in terms of individual processing steps – "packtivities" – linked through a dynamic directed acyclic graph (DAG) and present an initial set of JSON schemas for such a description and an implementation – "yadage" – capable of executing workflows of analysis preserved via Linux containers.

GitHub, Inc.

yadage - yaml based adage

DOI 10.5281/zenodo.309288 | pypi package 0.10.8 | build passing | health 94% | coverage 90% | docs latest | version git-master

A declarative way to define adage workflows using a JSON schema (but we'll always write it as YAML)

```
docker run --rm -it -v /var/run/docker.sock:/var/run/docker.sock -v $PWD:$PWD -w $PWD lukasheinrich/yada
yadage-run -t from-github/phenochain mdwork madgraph_delphes.yml -p nevents=100
```

or just

```
eval "$(curl https://raw.githubusercontent.com/diana-hep/yadage/master/yadagedocker.sh)"
yadage-run -t from-github/phenochain mdwork madgraph_delphes.yml -p nevents=100
```

This package reads and executes workflows adhering to the workflow JSON schemas defined at https://github.com/diana-hep/cap-schemas such as the onces stored in the community repository https://github.com/lukasheinrich/yadage-workflows. For executing the individual steps it mainly uses the packtivity python bindings provided by https://github.com/diana-hep/packtivity.

Possible Backends:

Yadage can run on various backends such as multiprocessing pools, ipython clusters, or celery clusters. If human intervention is needed for certain steps, it can also be run interactively.

Example Workflow

```
stages:
  - name: hello_world
```

reana.readthedocs.io

Fork me on GitHub

REANA - Reusable Analyses

build passing | coverage 100% | docs latest | Issues ready for work 2 | gitter join chat | license GNU General Public License v2.0

REANA is a system that permits to instantiate research data analyses on the cloud. It uses container-based technologies and was born to target the use case of particle physics analyses in LHC collaborations. The system paves the way to reusing and reinterpreting preserved data analyses even several years after the original analysis.

REANA - Reusable Analyses

Navigation

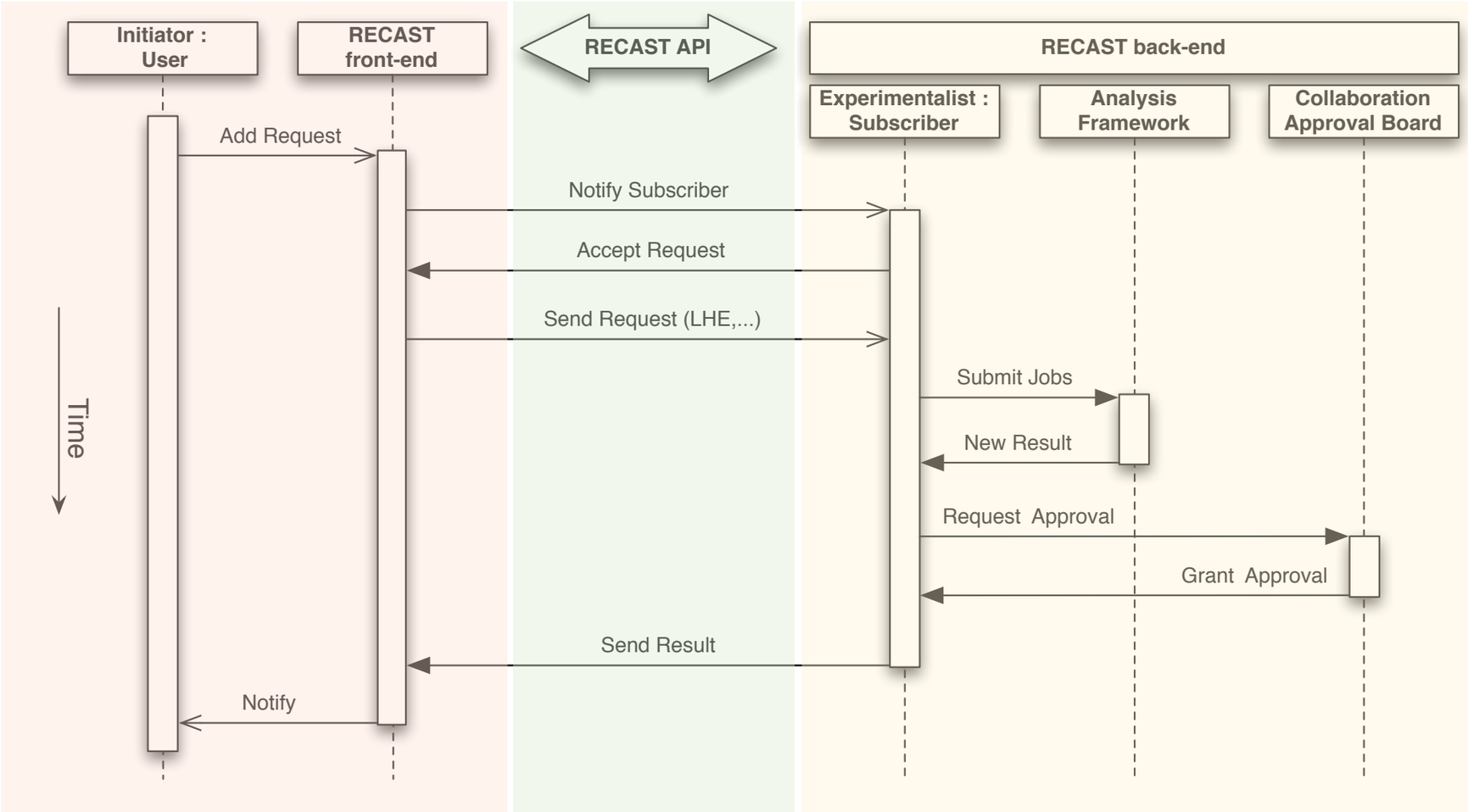
REANA@DockerHub
REANA@GitHub

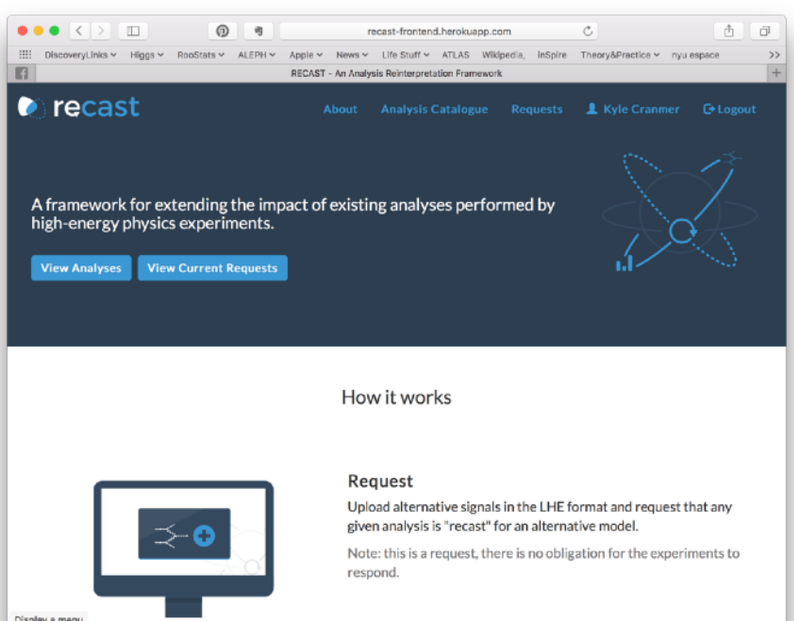
Quick search

Go

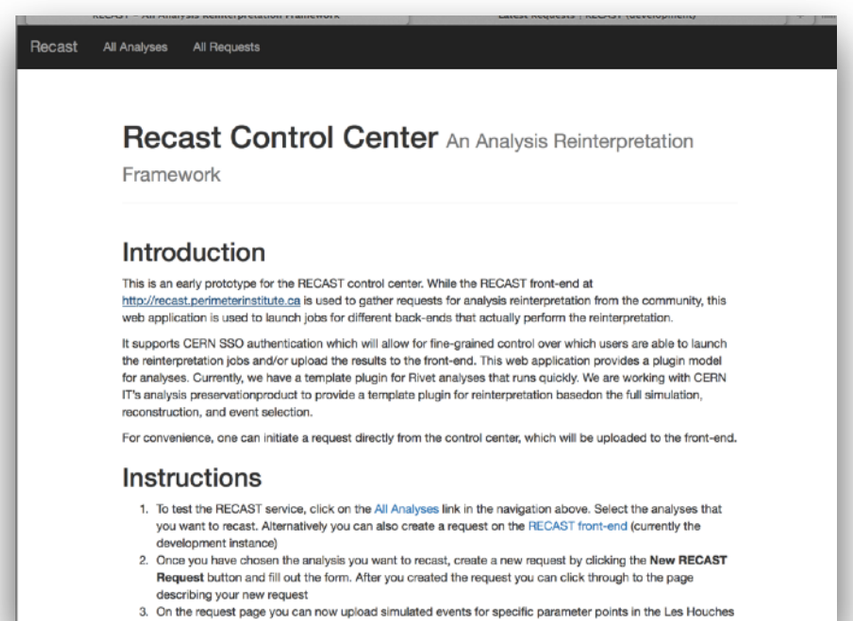
v: latest

https://arxiv.org/pdf/1706.01878.pdf

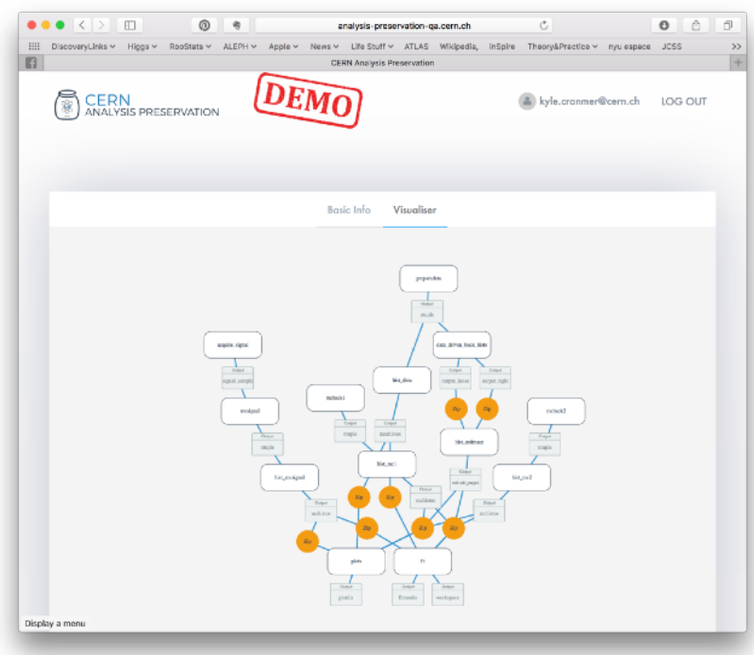




Front-End: public facing
collects requests



Control Center: not public, uses CERN auth.,
oversees processing of jobs on back-end



CERN Analysis Preservation:
Stores workflows, provides back-end
computing resources

Putting it all together

https://github.com/cranmer/active_sciencing

SYNTHESIS

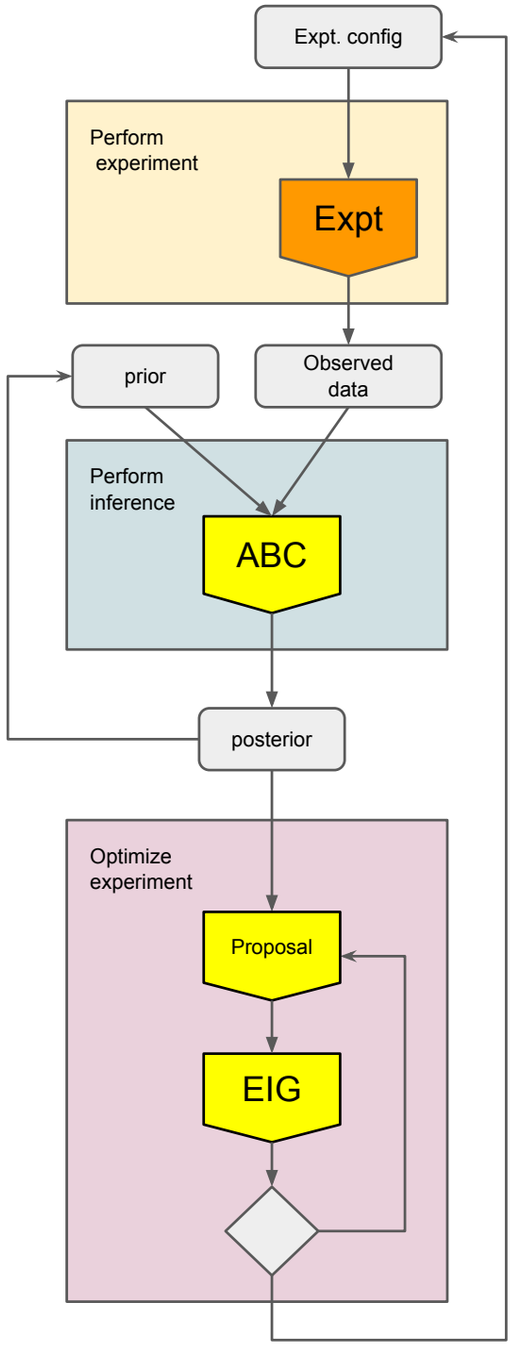
active learning / sequential design / black box optimization

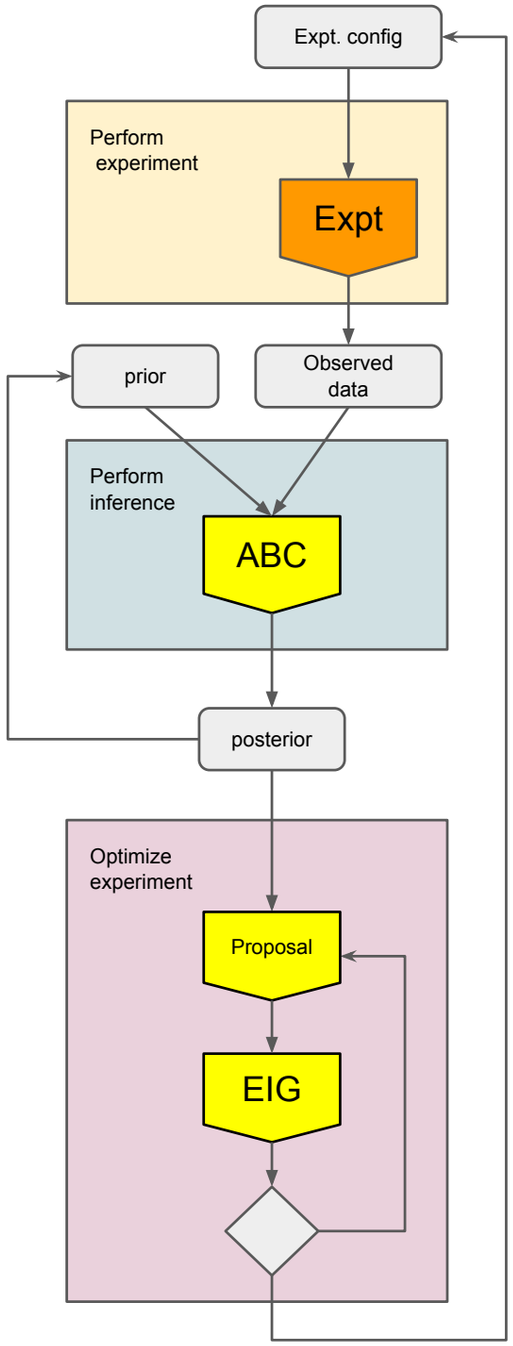
Active Sciencing

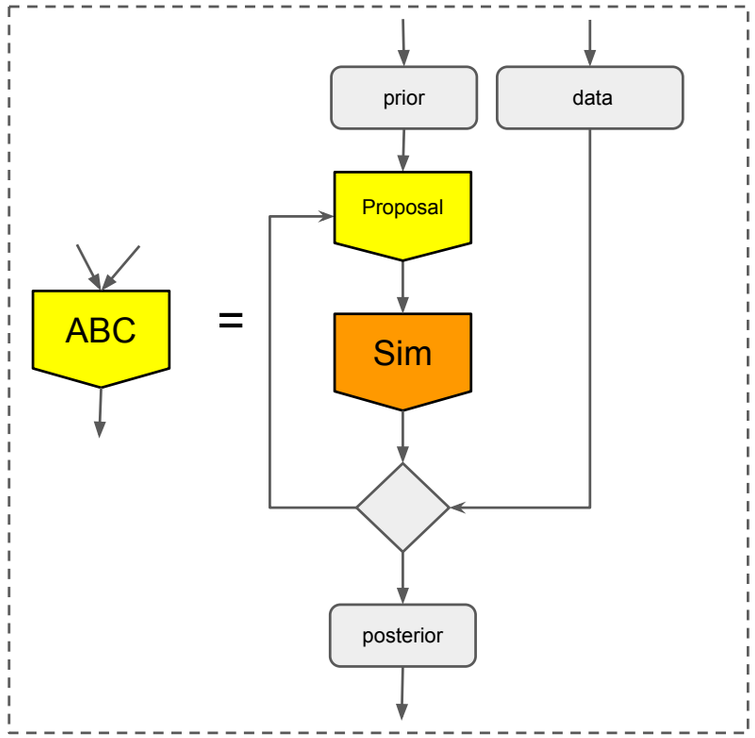
reusable workflows

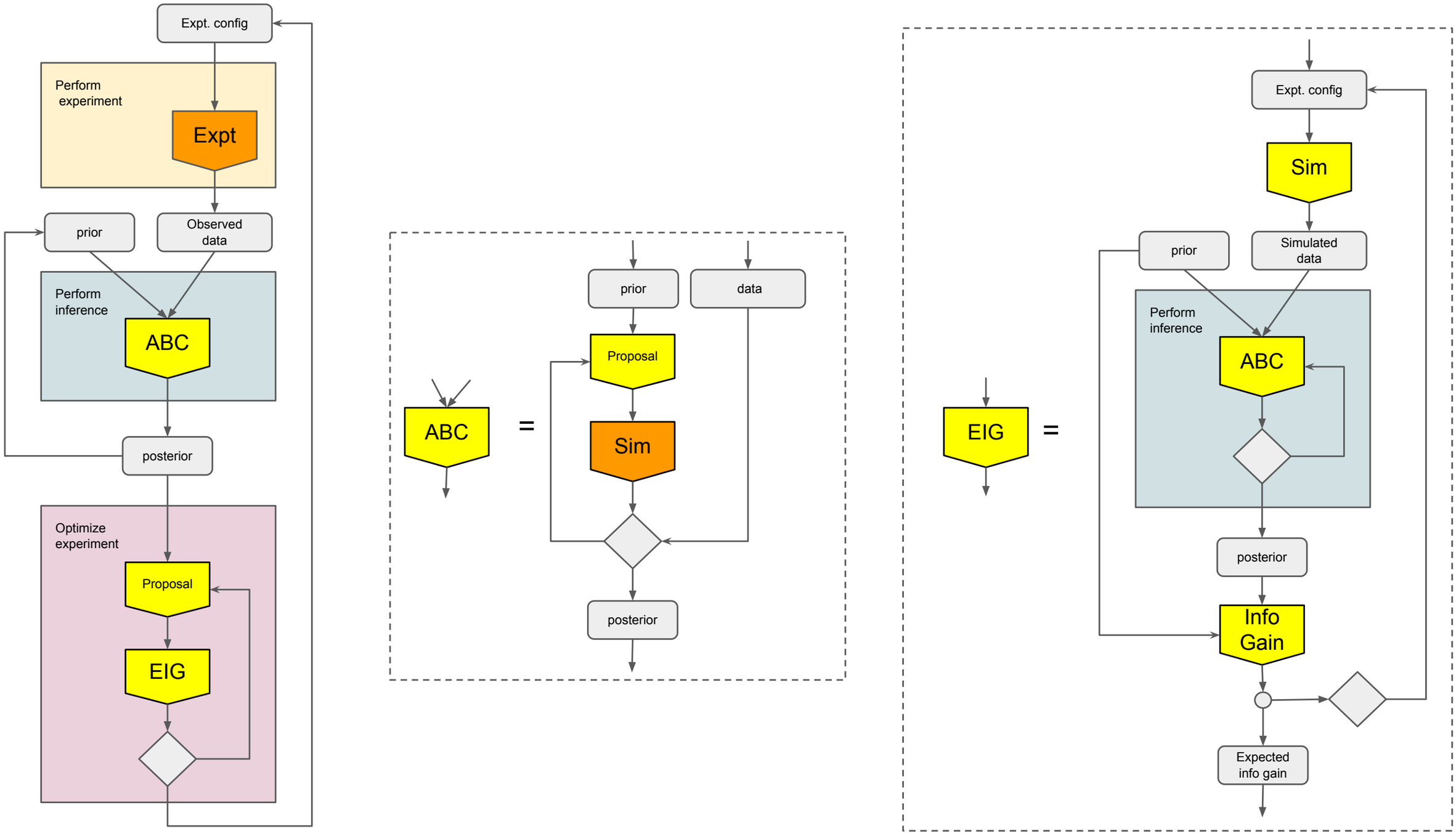
simulation-based
inference engines

https://github.com/cranmer/active_sciencing


Expt. config
Perform experiment
Expt
prior
Observed data
Perform inference
ABC
posterior
Optimize experiment
Proposal
EIG


Expt. config
Perform experiment
Expt
prior
Observed data
Perform inference
ABC
posterior
Optimize experiment
Proposal
EIG


ABC
=
prior
data
Proposal
Sim
posterior

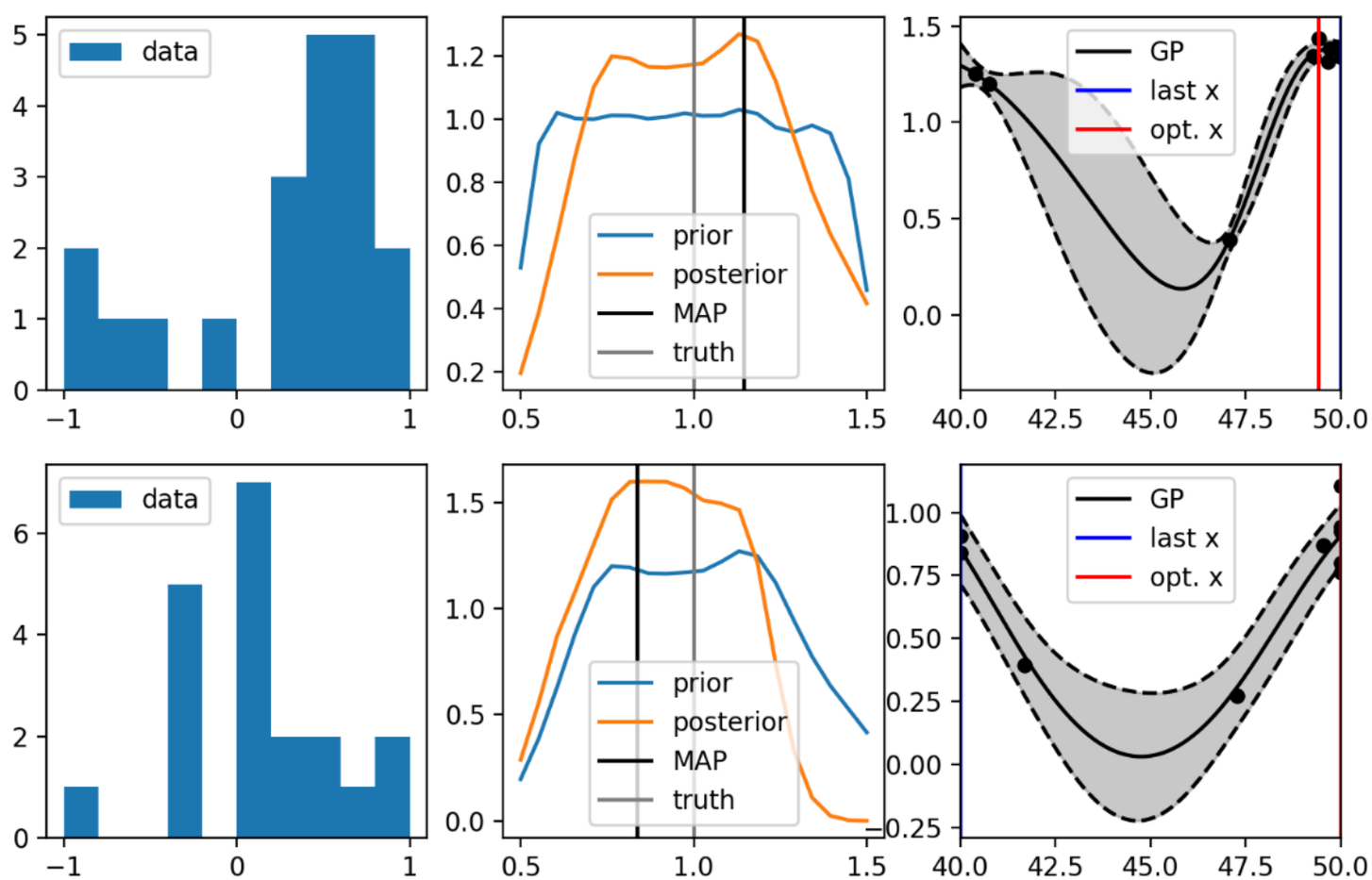

Expt. config
Perform experiment
Expt
prior
Observed data
Perform inference
ABC
posterior
Optimize experiment
Proposal
EIG
ABC
=
prior
data
Proposal
Sim
posterior
EIG
=
Expt. config
Sim
prior
Simulated data
Perform inference
ABC
posterior
Info Gain
Expected info gain

ACTIVE SCIENCING DEMO

Input:

- workflow for performing "real" experiment that returns data
- workflow for running simulator given parameters of theory and experimental configuration

Demo shows use of likelihood-free inference technique & Bayesian Optimization to measure the Weinberg angle and optimize beam energy (eg. just above or below $M_Z/2$)



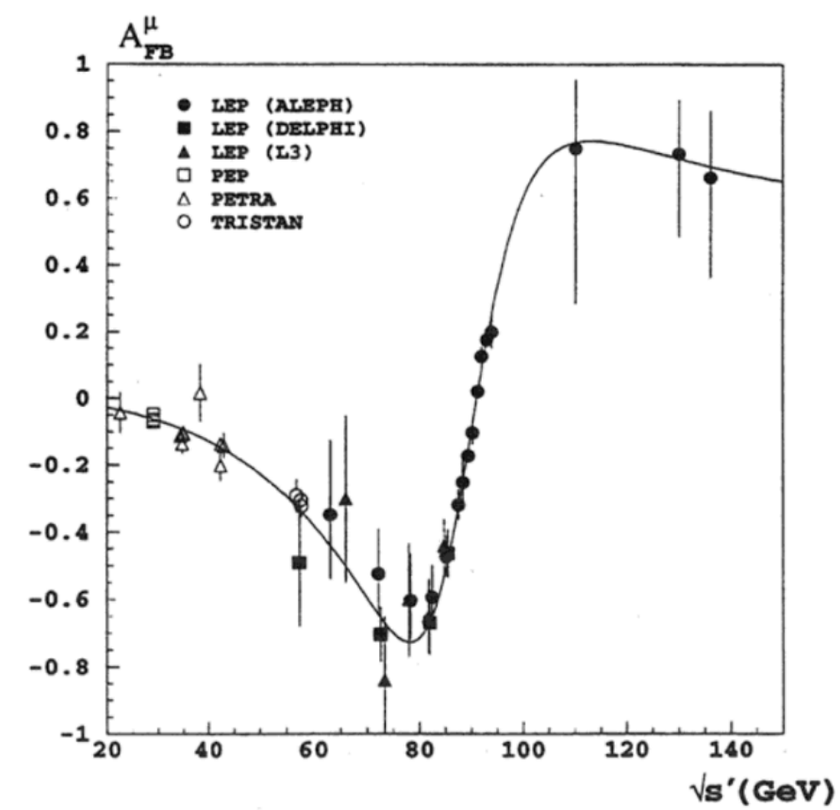


Figure 2: Measured forward-backward asymmetries of muon-pair production compared with the model independent fit results.

ENCAPSULATING THE SIMULATION

https://github.com/lukasheinrich/weinberg-test

README.md

Run HEP workflows from the web.

by Kyle Cranmer and Lukas Heinrich

An example notebook on how to generate simulated high energy physics collision events using the generator package MadGraph. Simulated datasets obtained from this notebook can then be used to train and evaluate the performance of generative models for physics.

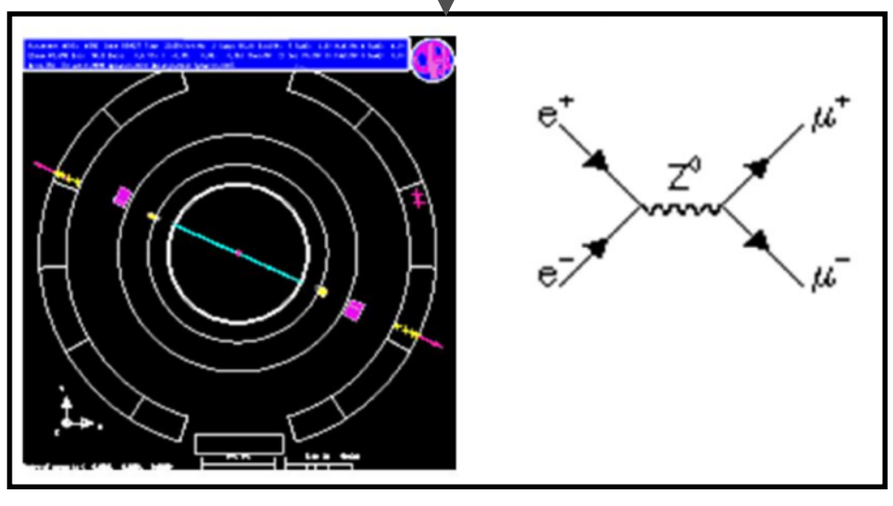
Usage:

This repository has been equipped with a Dockerfile to encapsulate its software environment. It can be used with the mybinder service to launch an ephemeral jupyter notebook server to run the notebook.

Click on the below badge and open the notebook `adage.ipynb`.

launch binder

$$
\begin{aligned}
\mathcal{L}_{SM} = & \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^a_{\mu\nu}G^{\mu\nu}_a}_{\text{kinetic energies and self-interactions of the gauge bosons}} \\
& + \underbrace{\bar{L}\gamma^\mu(i\partial_\mu - \frac{1}{2}g\tau\cdot\mathbf{W}_\mu - \frac{1}{2}g'YB_\mu)L + \bar{R}\gamma^\mu(i\partial_\mu - \frac{1}{2}g'YB_\mu)R}_{\text{kinetic energies and electroweak interactions of fermions}} \\
& + \underbrace{\frac{1}{2}\left|(i\partial_\mu - \frac{1}{2}g\tau\cdot\mathbf{W}_\mu - \frac{1}{2}g'YB_\mu)\,\phi\right|^2 - V(\phi)}_{W^\pm, Z, \gamma, \text{and Higgs masses and couplings}} \\
& + \underbrace{g''(\bar{q}\gamma^\mu T_a q)\,G^a_\mu}_{\text{interactions between quarks and gluons}} + \underbrace{(G_1\bar{L}\phi R + G_2\bar{L}\phi_c R + h.c.)}_{\text{fermion masses and couplings to Higgs}}
\end{aligned}
$$



other electroweak parameters. This can be shown with Eq. (2.96), giving

$$
A^f_{FB}(s) \simeq A^f_{FB}(m_Z^2) + \frac{(s - m_Z^2)}{s}\,\frac{3\pi\alpha(s)}{\sqrt{2}G_F m_Z^2}\,\frac{2Q_e Q_f g_{Ae} g_{Af}}{(g^2_{Ve} + g^2_{Ae})(g^2_{Vf} + g^2_{Af})} \quad . \quad (8.30)
$$

ENCAPSULATING THE SIMULATION

https://github.com/lukasheinrich/weinberg-test

README.md

Run HEP workflows from the web.

by Kyle Cranmer and Lukas Heinrich

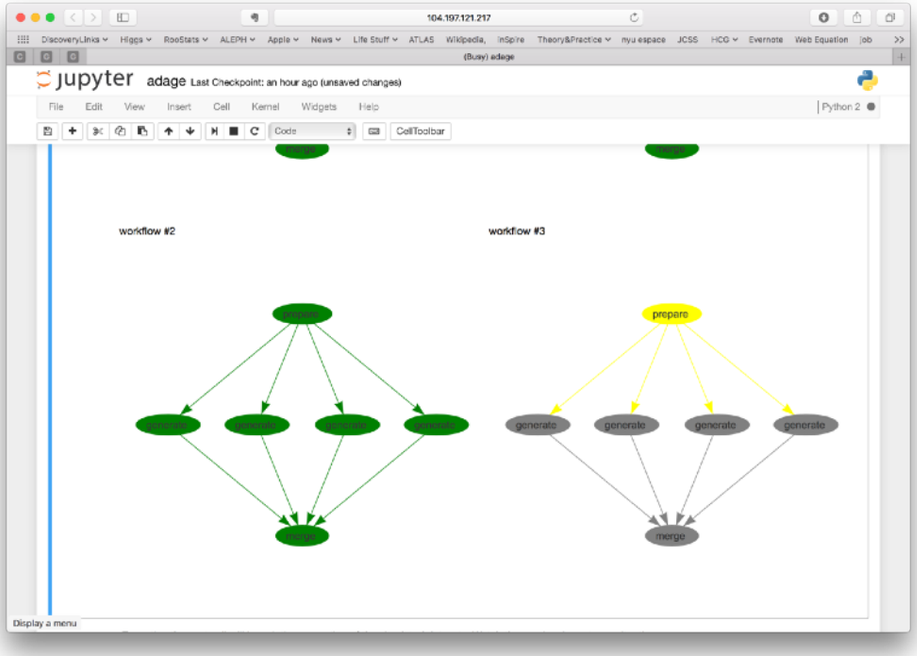
An example notebook on how to generate simulated high energy physics collision events using the generator package MadGraph. Simulated datasets obtained from this notebook can then be used to train and evaluate the performance of generative models for physics.

Usage:

This repository has been equipped with a Dockerfile to encapsulate its software environment. It can be used with the mybinder service to launch an ephemeral jupyter notebook server to run the notebook.

Click on the below badge and open the notebook `adage.ipynb`.

launch binder

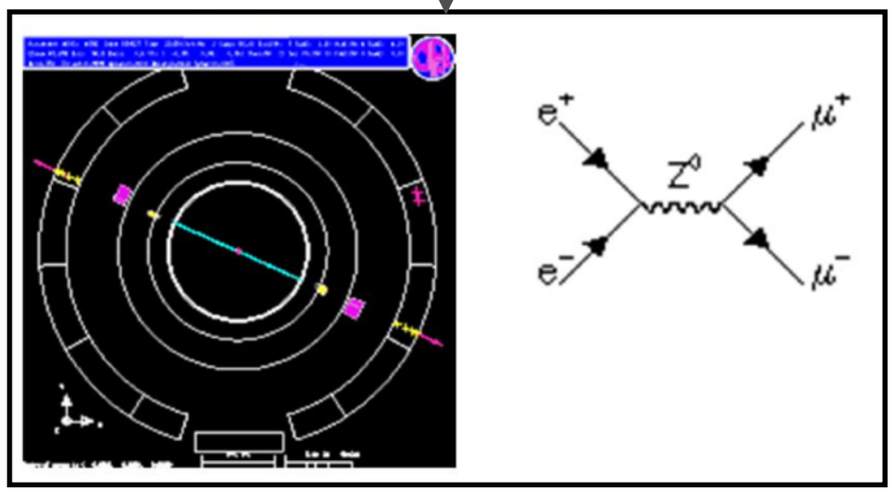


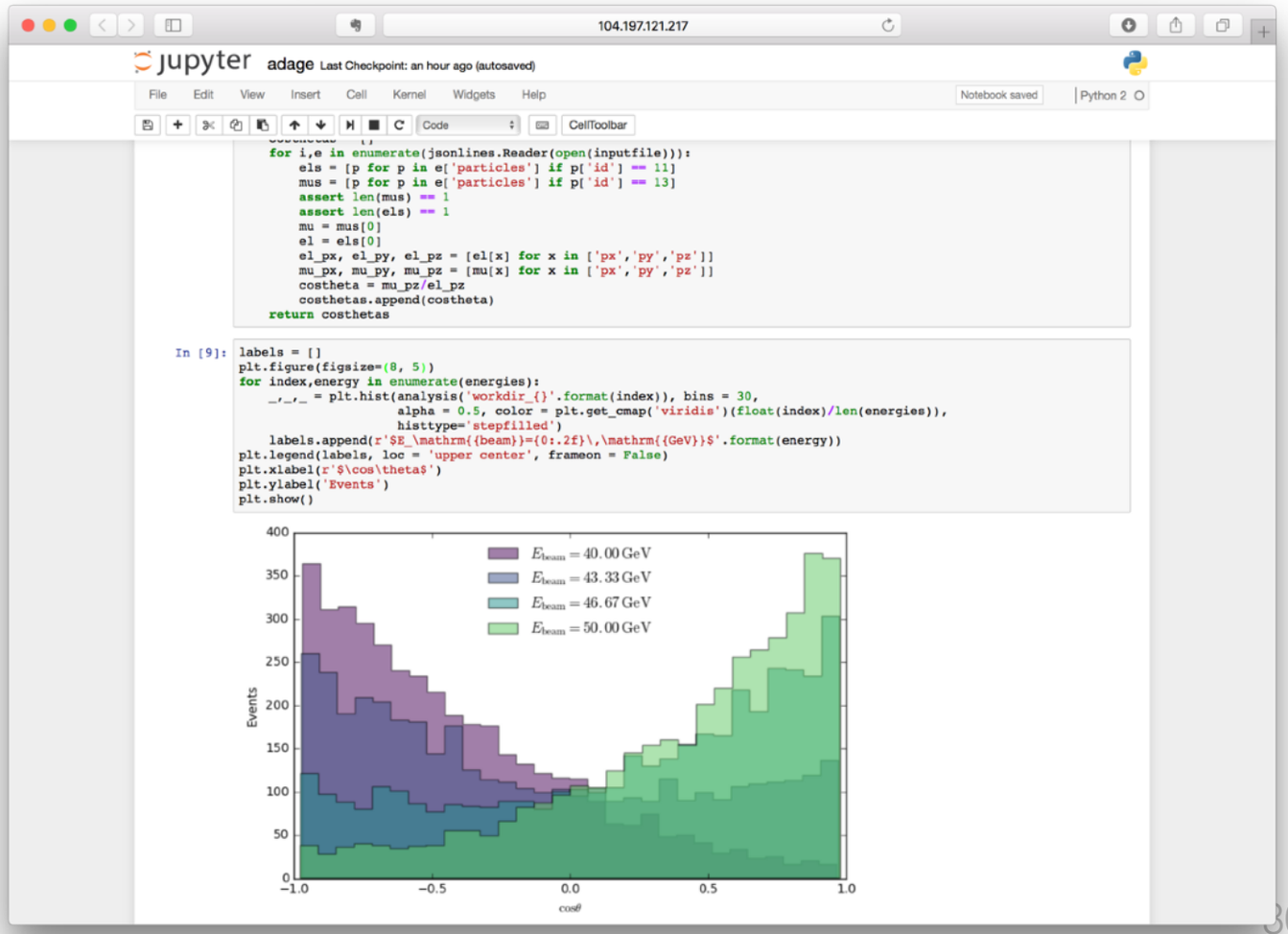
$$\mathcal{L}_{SM} = \underbrace{\frac{1}{4}\mathbf{W}_{\mu\nu}\cdot\mathbf{W}^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G^a_{\mu\nu}G^{\mu\nu}_a}_{\text{kinetic energies and self-interactions of the gauge bosons}}$$

$$+ \underbrace{\bar{L}\gamma^\mu(i\partial_\mu - \frac{1}{2}g\tau\cdot\mathbf{W}_\mu - \frac{1}{2}g'YB_\mu)L + \bar{R}\gamma^\mu(i\partial_\mu - \frac{1}{2}g'YB_\mu)R}_{\text{kinetic energies and electroweak interactions of fermions}}$$

$$+ \underbrace{\frac{1}{2}\left|(i\partial_\mu - \frac{1}{2}g\tau\cdot\mathbf{W}_\mu - \frac{1}{2}g'YB_\mu)\,\phi\right|^2 - V(\phi)}_{W^\pm, Z, \gamma, \text{and Higgs masses and couplings}}$$

$$+ \underbrace{g''(\bar{q}\gamma^\mu T_a q)\,G^a_\mu}_{\text{interactions between quarks and gluons}} + \underbrace{(G_1\bar{L}\phi R + G_2\bar{L}\phi_c R + h.c.)}_{\text{fermion masses and couplings to Higgs}}$$

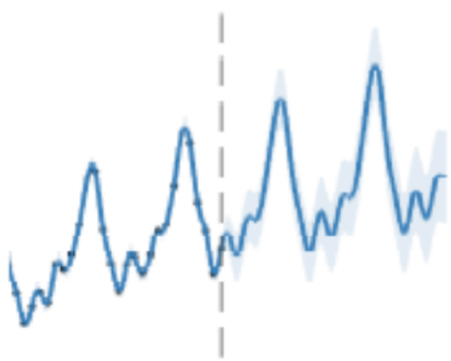




SEARCHING OVER SPACE OF MODELS

Vocabulary of kernels + grammar for composition

- physics goes into the construction of a "Kernel" that describes covariance of data

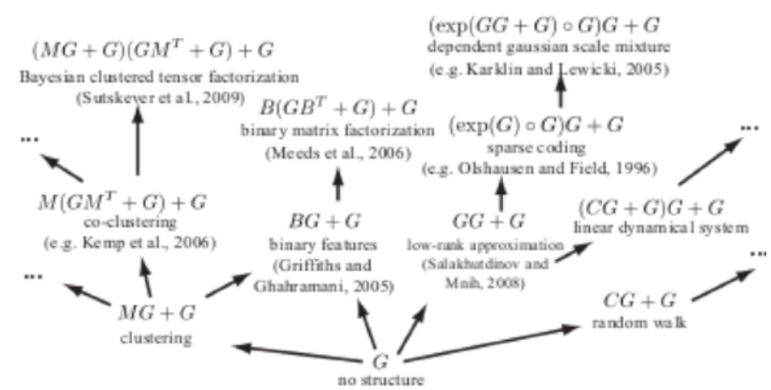


Structure Discovery in Nonparametric Regression through Compositional Kernel Search

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum, Zoubin Ghahramani

International Conference on Machine Learning, 2013

pdf | code | poster | bibtex



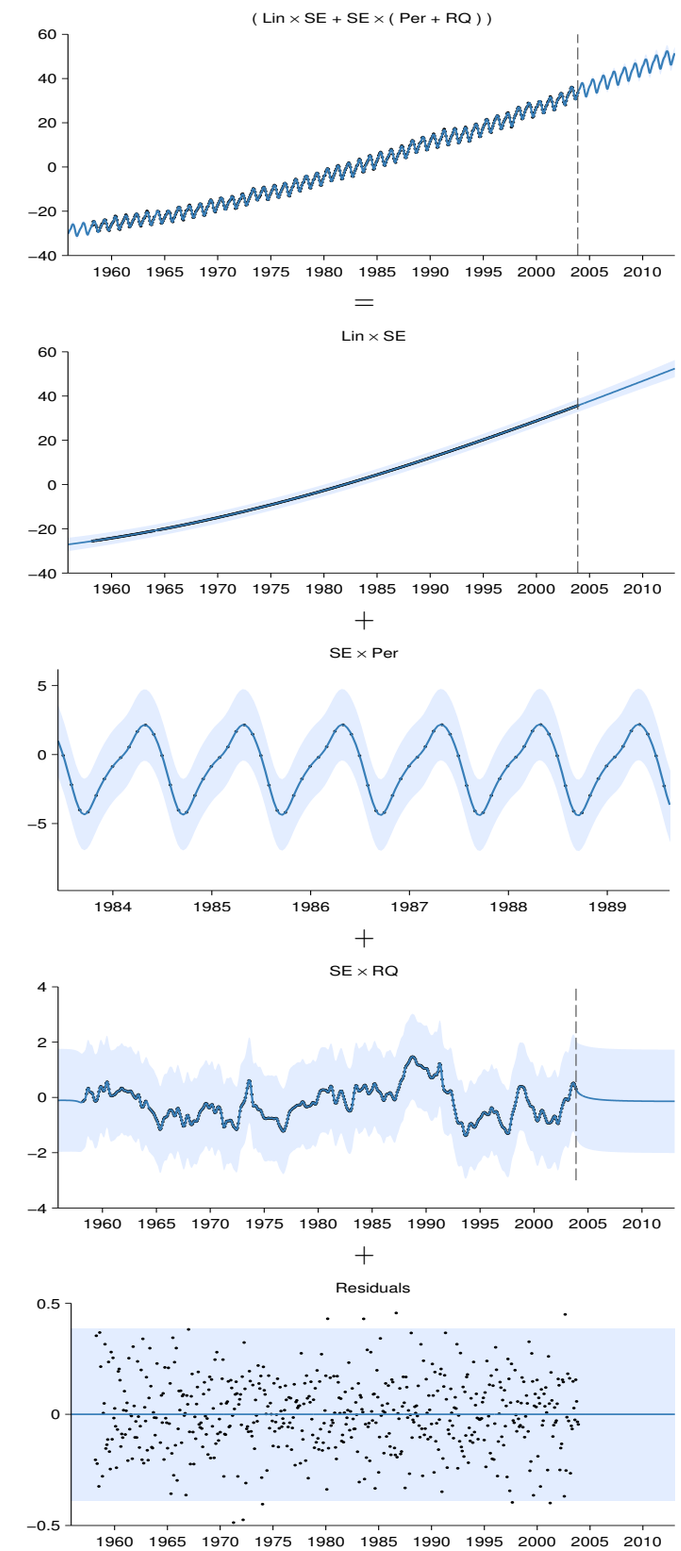
Exploiting compositionality to explore a large space of model structures

Roger Grosse, Ruslan Salakhutdinov, William T. Freeman, Joshua B. Tenenbaum

Conference on Uncertainty in Artificial Intelligence, 2012

pdf | code | bibtex

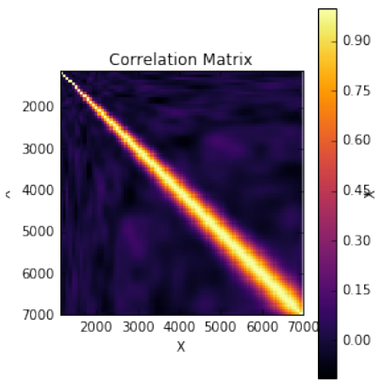
Mauna Loa atmospheric CO_2



GAUSSIAN PROCESSES AT LHC

with Meghan Frate

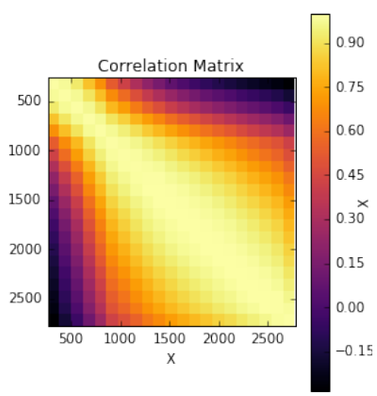
Instead of fitting the dijet spectrum with an ad hoc 3-5 parameter function, use GP with kernel motivated from physics



Final Kernel =

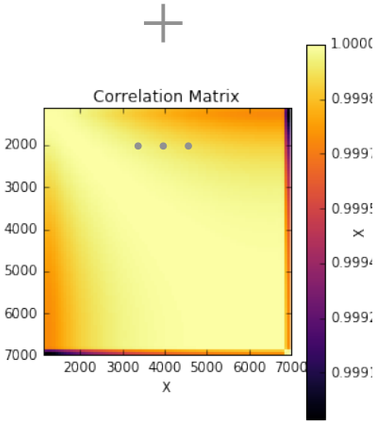
Poisson stats
+ Mass Resolution

=



+ Parton Density Functions

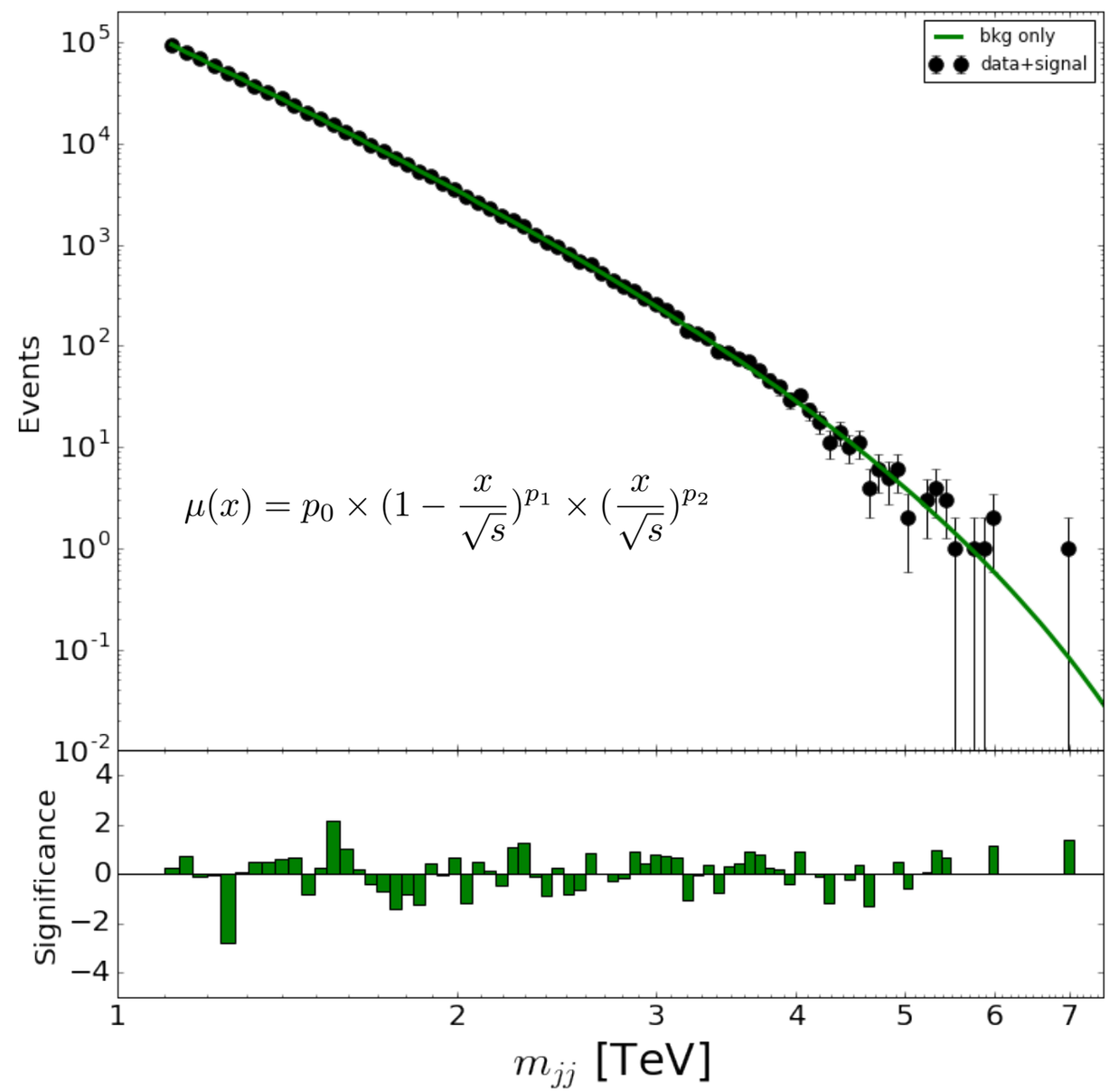
+



+ Jet Energy Scale

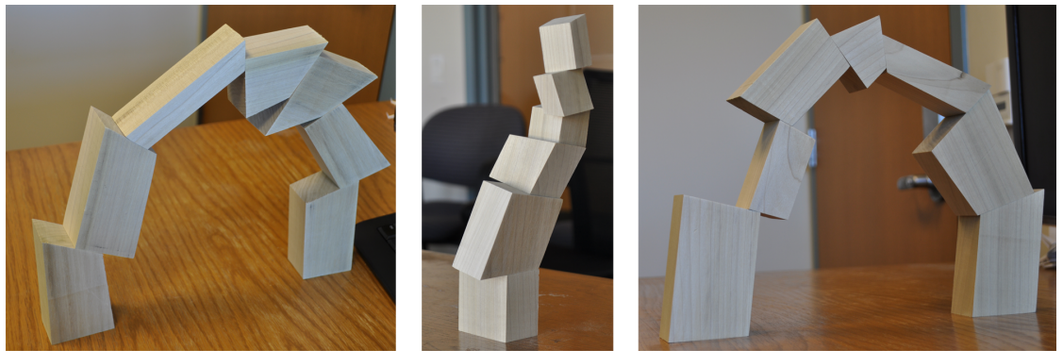
+

. . .

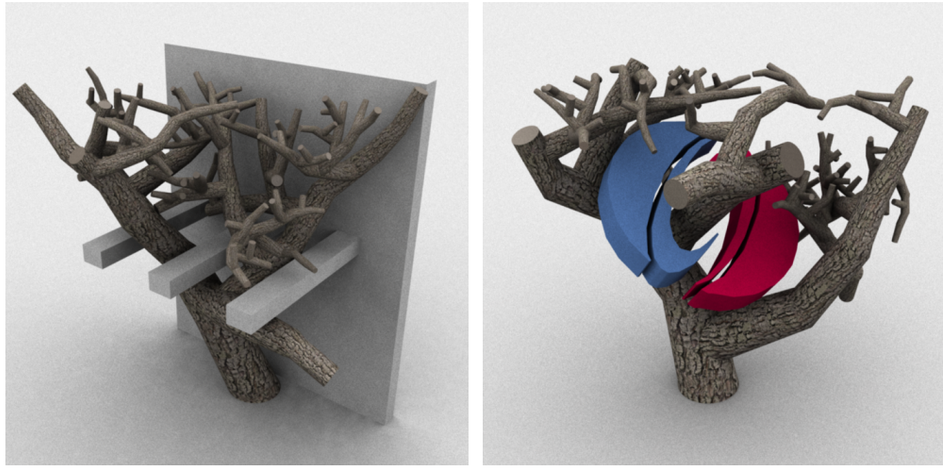


Directed Procedural Graphics

Stable Static Structures



Procedural Graphics



$\mathbf{x}$	$\mathbf{y}$
simulation	constraint

Ritchie, Lin, Goodman, & Hanrahan.
Generating Design Suggestions under Tight Constraints
with Gradient-based Probabilistic Programming.
In Computer Graphics Forum, (2015)

Ritchie, Mildenhall, Goodman, & Hanrahan.
"Controlling Procedural Modeling Programs with
Stochastically-Ordered Sequential Monte Carlo."
SIGGRAPH (2015)

COLLABORATORS

Gilles Louppe

Kyunghyun Cho

Joan Bruna

Daniel Whiteson

Michael Kagan

Juan Pavez

THANKS!

Physicists:
Lukas Heinrich,
Cyril Becot,
Daniel Whiteson,
Michael Kagan,
Johann Brehmer,
Taylor Faucett
David Rousseau

ML/AI:
Gilles Louppe,
Juan Pavez,
Kyunghyun Cho,
Brenden Lake,
Pierre Baldi,
Peter Sadowski,
Uri Shalit,
Joan Bruna,
Yann LeCun,
Balázs Kégl
Cecile Germain

NYU CDS Masters Students
Xinyi Gong,
Zihao Wang,
Lanyu Shang,
Alex Pine,
Israel Malkin,
Charlie Guthrie,
Manoj Kumar,
Phil Yeres,
Michele Ceru
Hao Liu,
Li Ke,
Yuhao Zhao

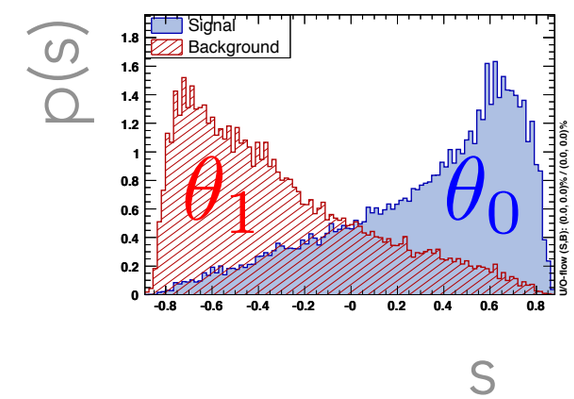
CARL

The intractable likelihood ratio based on high-dimensional features x is:

$$\frac{p(x|\theta_0)}{p(x|\theta_1)}$$

We can show that an **equivalent test** can be made from 1-D projection

$$\frac{p(x|\theta_0)}{p(x|\theta_1)} = \frac{p(s(x;\theta_0,\theta_1)|\theta_0)}{p(s(x;\theta_0,\theta_1)|\theta_1)}$$



if the scalar map s: X → $\mathbb{R}$ has the same level sets as the likelihood ratio

$$s(x;\theta_0;\theta_1) = \text{monotonic}[\ p(x|\theta_0)/p(x|\theta_1)\ ]$$

Estimating the density of $s(x;\theta_0,\theta_1)$ via the simulator calibrates the ratio.

K.C., G. Louppe, J. Pavez: http://arxiv.org/abs/1506.02169

Binary classifier on balanced y=0 and y=1 labels learns

$$s(x) = \frac{p(x|y=1)}{p(x|y=0) + p(x|y=1)}$$

Which is one-to-one with the likelihood ratio

$$\frac{p(x|y=0)}{p(x|y=1)} = 1 - \frac{1}{s(x)}$$

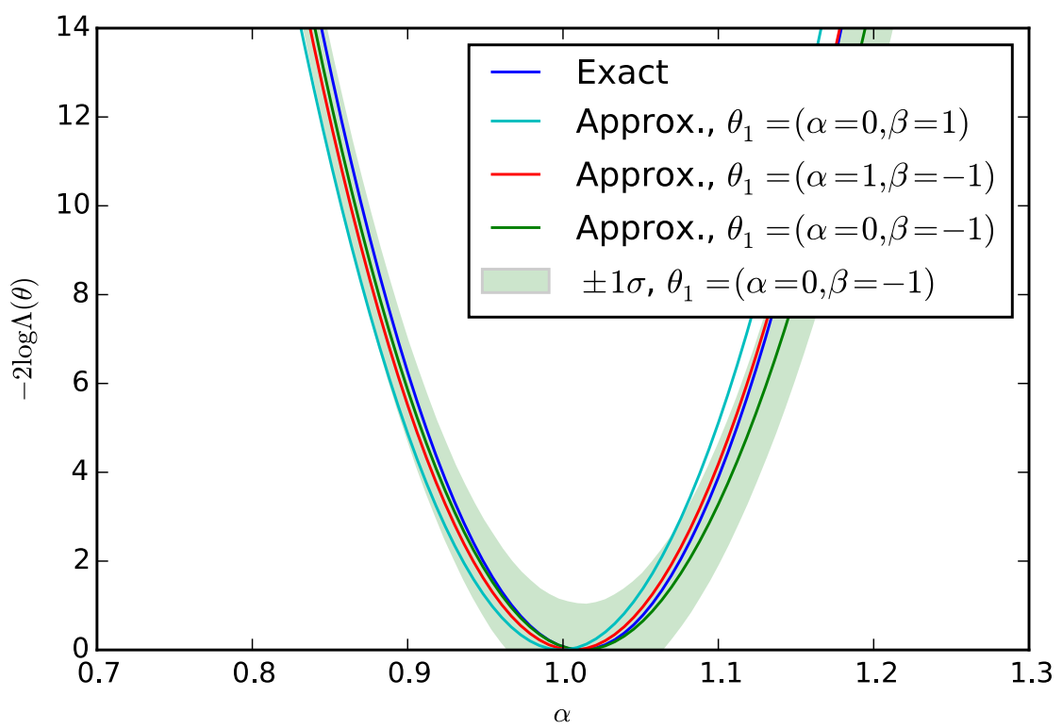
Can do the same thing for any two points θ_0 & θ_1 in parameter space. I call this a **parametrized classifier**

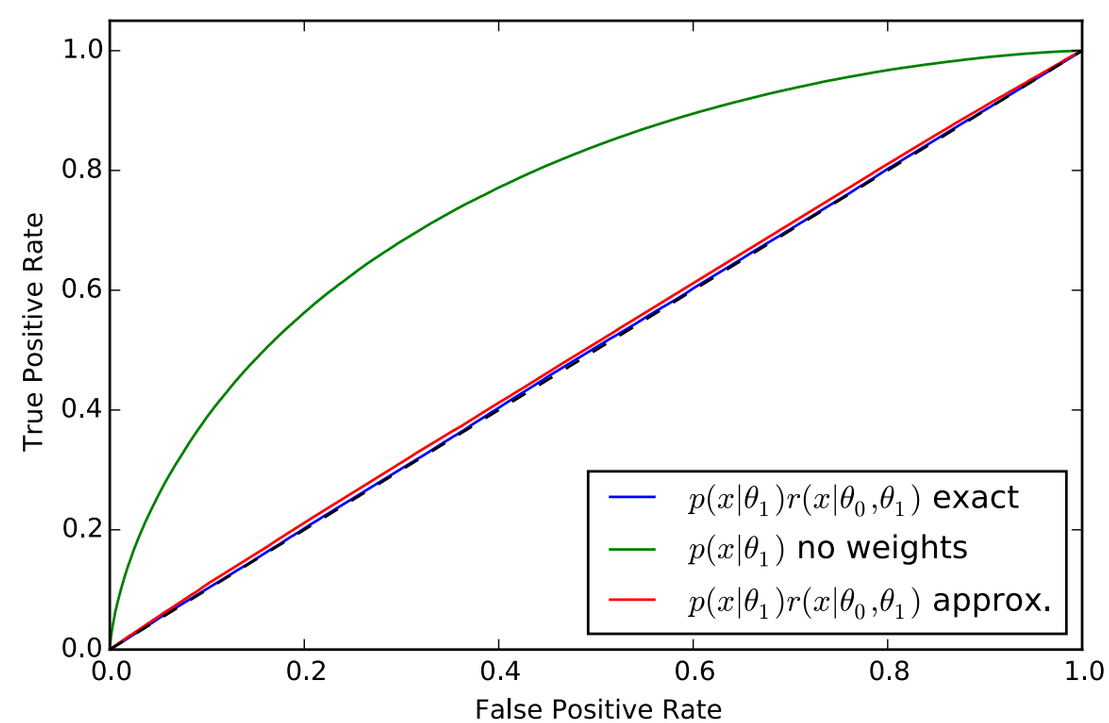
$$s(x; \theta_0, \theta_1) = \frac{p(x|\theta_1)}{p(x|\theta_0) + p(x|\theta_1)}$$

DIAGNOSTICS

In practice $\hat{r}(\hat{s}(\mathbf{x};\theta_0,\theta_1))$ will not be exact. Diagnostic procedures are needed to assess the quality of this approximation.

1. For inference, the value of the MLE $\hat{\theta}$ should be independent of the value of θ_1 used in the denominator of the ratio.
2. Train a classifier to distinguish between unweighted samples from $p(\mathbf{x}|\theta_0)$ and samples from $p(\mathbf{x}|\theta_1)$ weighted by $\hat{r}(\hat{s}(\mathbf{x};\theta_0,\theta_1))$.





$$\frac{p_1(s^*)}{p_0(s^*)} = \frac{p_1(x)}{p_0(x)} \boxed{\frac{\int d\Omega_{s^*} p_0(x)/|\hat{n}\cdot\nabla s|}{\int d\Omega_{s^*} p_0(x)/|\hat{n}\cdot\nabla s|}} = \frac{p_1(x)}{p_0(x)}$$