Differentiable Matrix Elements with MadJax

Lukas Heinrich, CERN
Michael Kagan, SLAC

Introduction

Matrix Elements Calculations are the bread and butter of high-energy physics

- **most closely connected to the hard-processes we are interested in**
- **exploiting more information from Matrix Elements always useful**

$$\sigma(z\,|\,\theta) \qquad \sigma(\theta) = \int \mathrm{dz}\ \sigma(z\,|\,\theta) \qquad p(x\,|\,\theta) = \frac{1}{\sigma(\theta)}\sigma(x\,|\,\theta)$$

This talk:

- **automatic calculation of Matrix Element Derivatives inspired by ML methods**

$$\nabla_z \sigma(z\,|\,\theta) = \sigma(z\,|\,\theta) \qquad \nabla_\theta \sigma(z\,|\,\theta) = \sigma(z\,|\,\theta)$$

phase-space gradients

theory-space gradients

Context: the rise of differentiable programming

Modern Deep Learning at its core is about gradients

- vast number of neural net architectures but all **optimized using gradient descent**

New focus on making computational science pipeline differentiable

- allows us to freely mix & match ML modules with physics modules
- towards "physics"-aware ML methods
- trend goes beyond HEP

1990

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

Jürgen Schmidhuber*
Institut für Informatik
Technische Universität München
Arcisstr. 21, 8000 München 2, Germany
schmidhu@tumult.informatik.tu-muenchen.de

Differentiable Programming in High-Energy Physics

(Oxford), Kyle Cranmer (NYU), Matthew Feickert (UIUC),
miLab), Lukas Heinrich (CERN), Alexander Held (NYU)
rbilt) Mark Neubauer (UIUC), Jannicke Pearkes (Stanford),
Lund), Nick Smith (FermiLab), Giordon Stark (UCSC),
n), Vassil Vassilev (Princeton), Gordon Watts (U. Washington)

August 31, 2020

Differentiable Simulators for HEP

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

[1]CERN
[2]SLAC National Accelerator Laboratory
[3]Colorado State University

of gradient-based optimization. Deep
uild a complex computational pipeline
ting such functions is enabled through
e success of deep learning has led to an
ing promoted to a first-class citizen in
often involves replacing some common

Differentiable biology: using deep learning for biophysics-based and data-driven modeling of molecular mechanisms

Mohammed AlQuraishi & Peter K. Sorger

Nature Methods **18**, 1169–1180 (2021) | Cite this article
6497 Accesses | **127** Altmetric | Metrics

Abstract Biophysics

Deep learning using neural networks relies on a class of machine-learnable models

HEP Statistics

pyhf
∂ifferentiable
ℒikelihoods

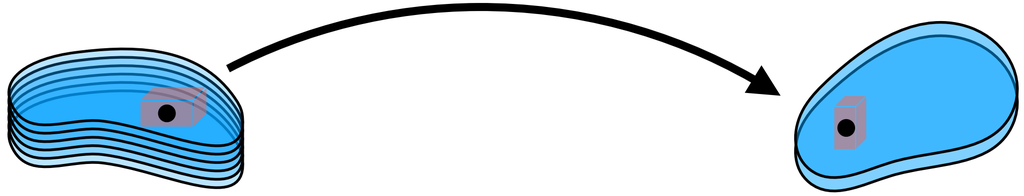
Beyond ML:

- gradients are useful in a lot of contexts
- but ML gave us new tools to compute gradients

Automatic Differentiation

AD is a technique to evaluate exact gradients without symbolic differentiation

- **in short: clever use of matrix-free techniques to evaluate Jacobians**



$$y = f(x)$$

$$dy = J_f \, dx$$

$$J_f = \frac{\partial(y_1, \ldots, y_m)}{\partial(x_1, \ldots, x_n)}$$

Most popular AD frameworks from ML usually in python:

TensorFlow PYTORCH

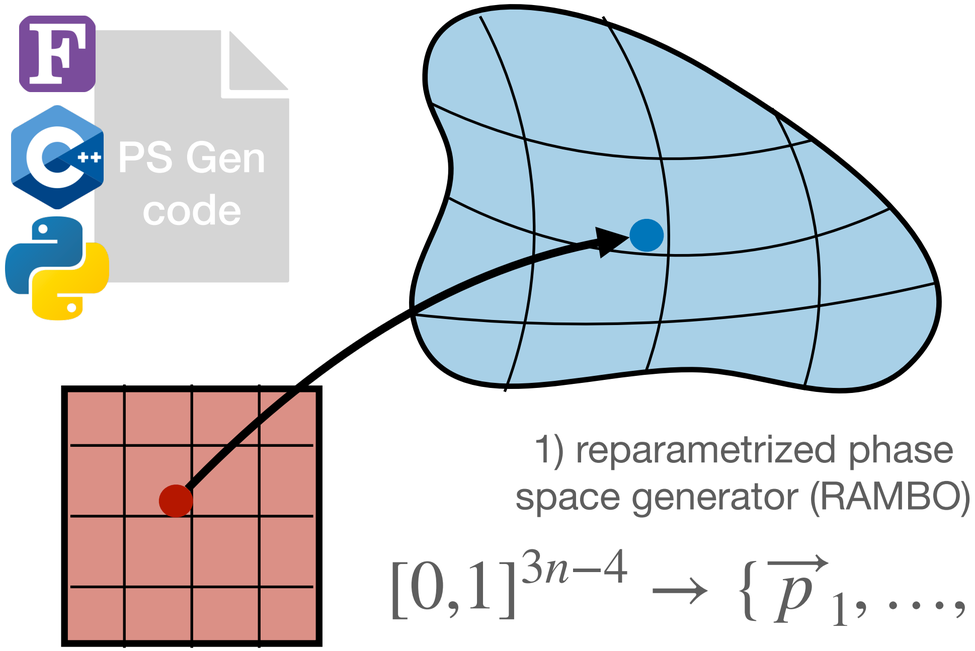
Often challenging to make code differentiable without a rewrite

- **turns out: not so much for MadGraph**

MadGraph

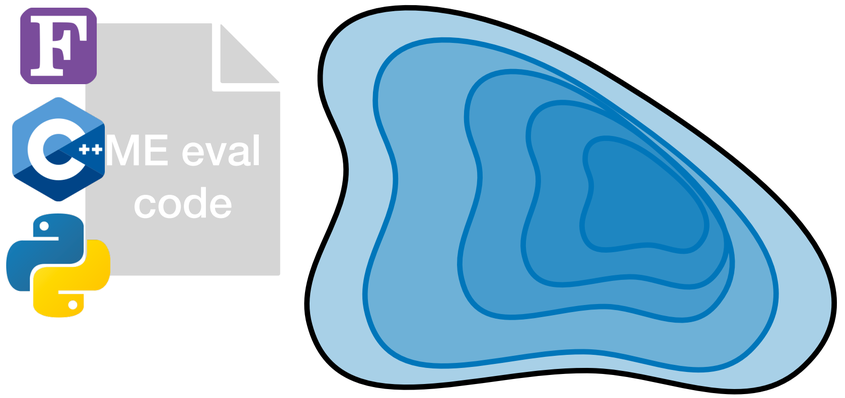
Disclaimer: we're not MadGraph experts, but the MG tooling made it easy for us

Idea: MadGraph is code-generator for Matrix-Element Code yielding code for



1) reparametrized phase space generator (RAMBO)

$$[0,1]^{3n-4} \to \{\vec{p}_1, \ldots, \vec{p}_n\}$$



2) matrix-element function

$$\mathrm{d}\sigma(x\,|\,\theta) = \mathrm{d}\sigma(\vec{p}_1, \ldots \vec{p}_n, |\,\theta)$$

MadGraph supports various languages, but no differentiability

Idea: add differentiable export option!

MadJax

MadJax adapts the existing Python Exporter (h/t V. Hirschi) to be differentiable

- **uses JAX as a AD backend**
- **solves edge cases that are not easily differentiable (root-finding conditionals, ...)**
- **JIT compilation for much faster evaluation than MG5 Python export**

Final Interface is simple:

```
pip install madjax

$> cat ee_to_mumu.mg5
generate mu+ mu- > e+ e-
output madjax ee_to_mumu

$> mg5_aMC
--mode=madjax_me_gen
-f ee_to_mumu.mg5
```

```
import madjax

mj = madjax.MadJax('ee_to_mumu')

random_variables = [0.2]*2
matrix_element = mj.matrix_element(
    E_cm=91.0, process_name='Matrix_1_mupmum_epem'
)

pars = { ("mass", 23): 9.918800e01, ("sminputs", 2): 1.166390e-05 }
value,gradients = matrix_element(pars, random_variables)
print(value)
gradients
```

✓ 6.2s

```
0.1099516438119396

{('mass', 23): DeviceArray(-0.01722685, dtype=float64),
 ('sminputs', 2): DeviceArray(6805.51017429, dtype=float64)}
```

MadJax

MadJax adapts the existing Python Exporter (h/t V. Hirschi) to be differentiable

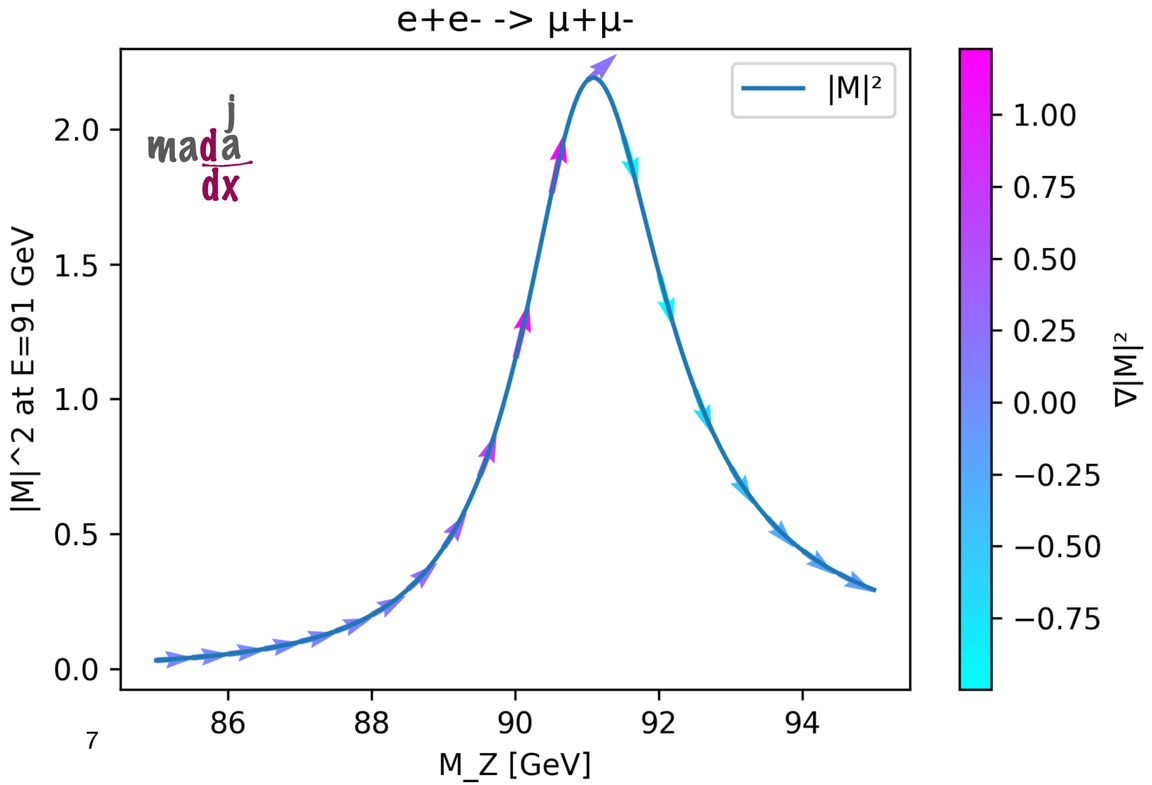
- **uses JAX as a automatic differentiation backend**
- **solves a number of edge cases that are not immediatly differentiable internal to wave-function and phase-space code (root-finding, conditionals etc)**
- **enables JIT compilation for much faster evaluation than existing Python export**

Final Interface is simple:

```
pip install madjax

# ee_to_mumu.mg5
generate mu+ mu- > e+ e-
output madjax ee_to_mumu

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



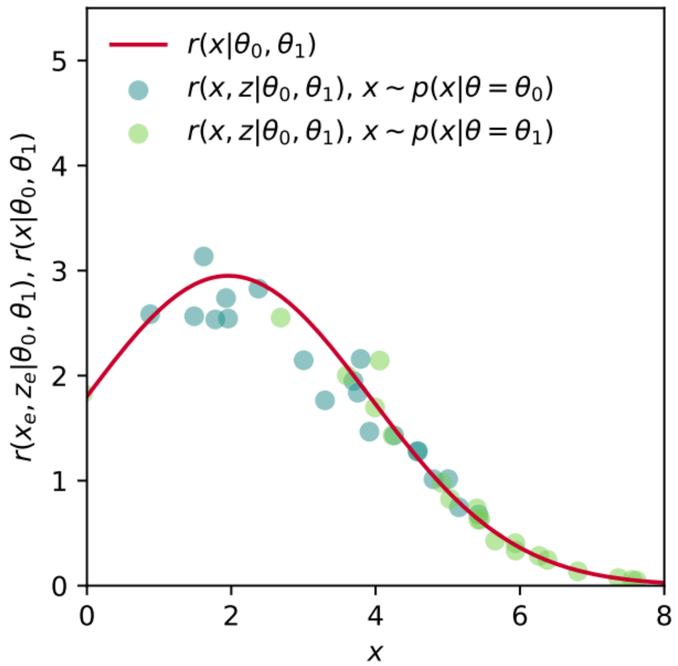
Example Use-Case I: Automatic Likelihood-Free Inference

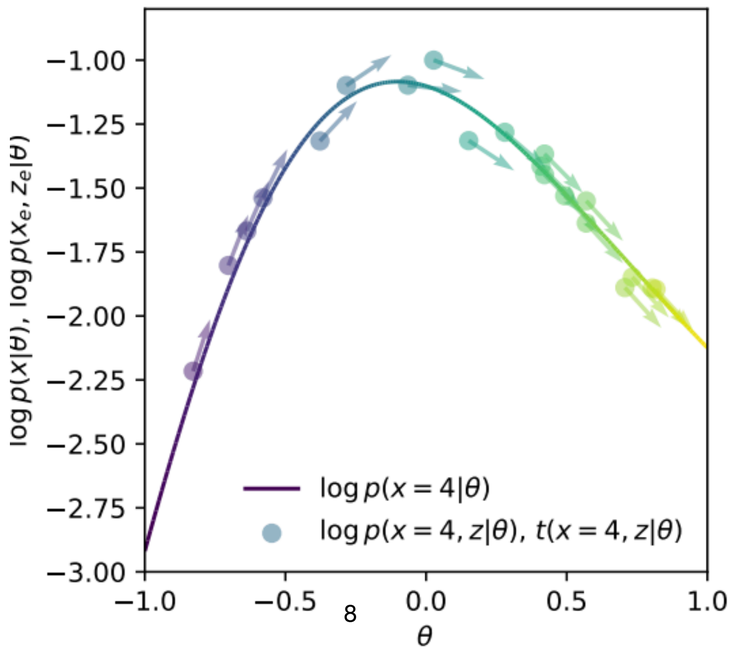
Mining Gold is a series of methods or "Likelihood-Free Inference" using joint quanties of the reconstructed (x) and truth-level event (z) as noisy training label

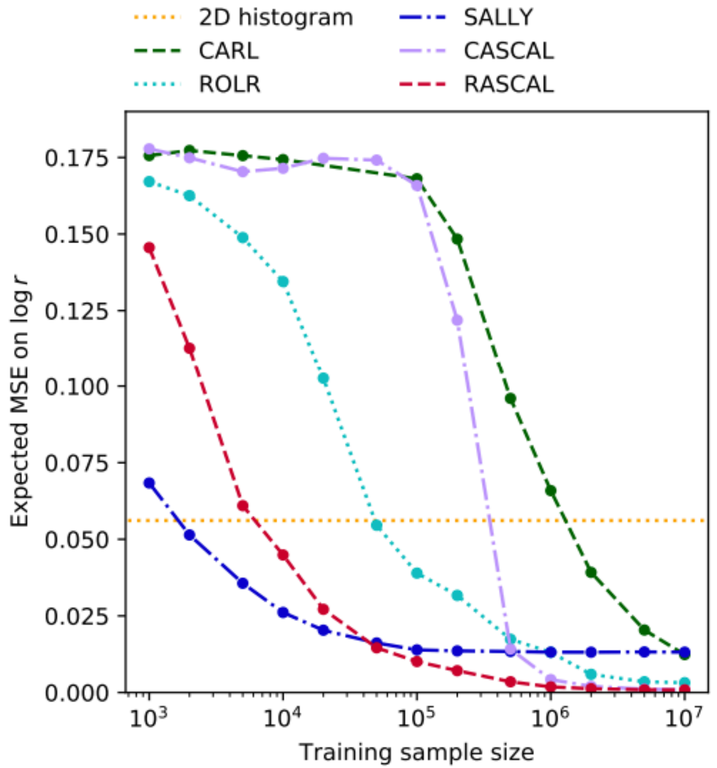
- **methods making use of gradient data are vastly more data-efficient**

$$r(x, z, \theta_0, \theta_1) = \frac{p(x, z \mid \theta_0)}{p(x, z \mid \theta_1)}$$

$$\nabla_\theta \log p(x, z \mid \theta)$$





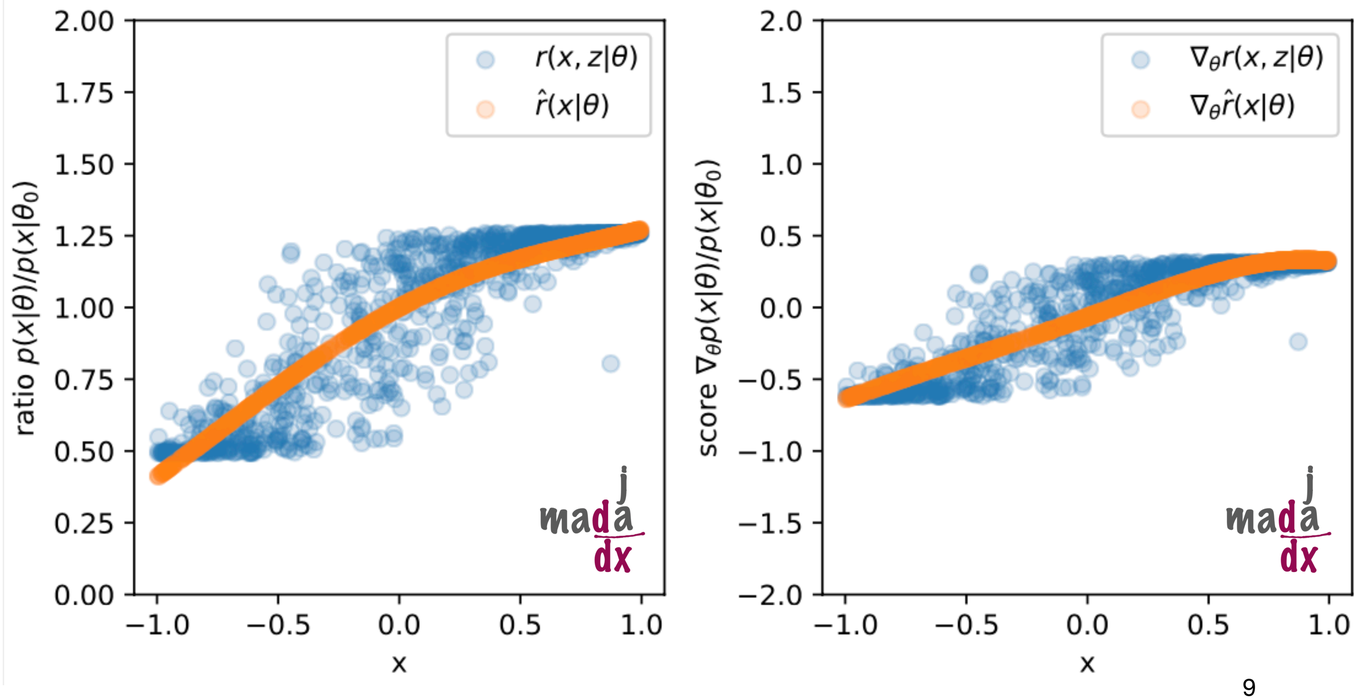


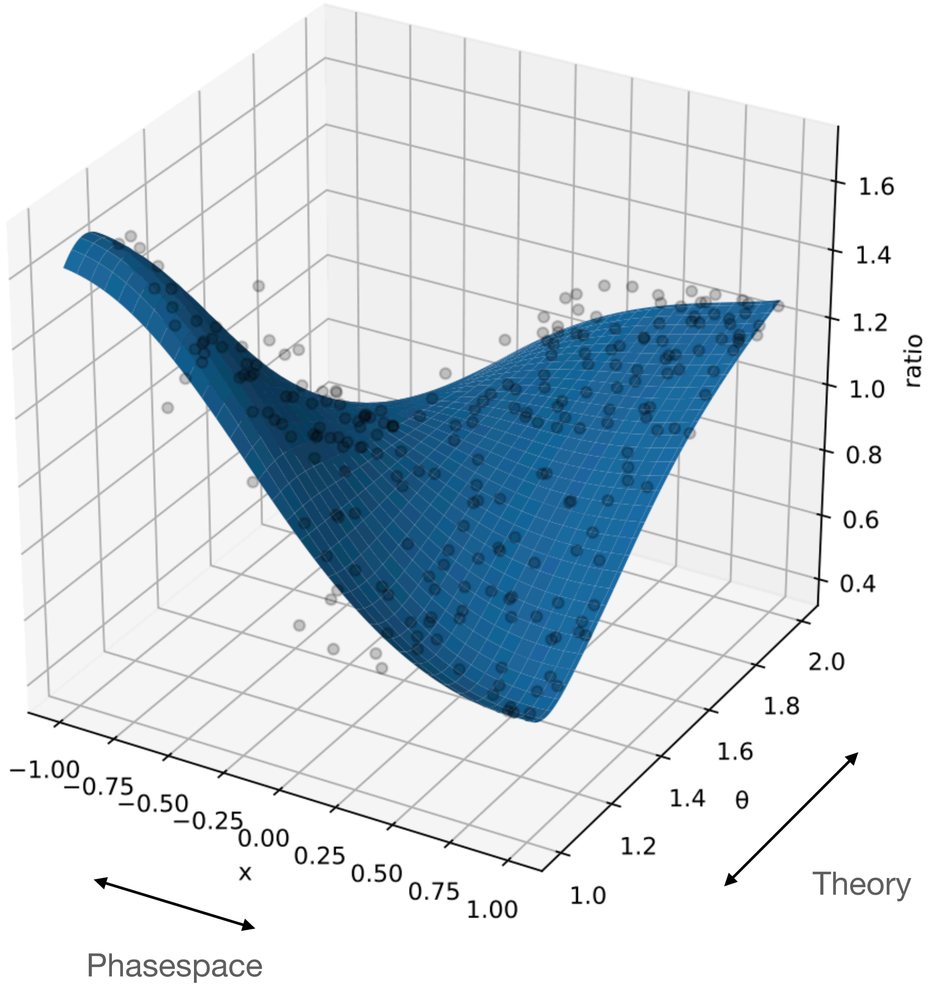
Automatic Likelihood-free Inference:

Current use-cases exploit analytic gradients (e.g. polynomial relations)

- **MadJax enables gradient-based "Mining Gold" for arbitrary theory parameters**
- **Toy Example: gradient-based likelihood ratio estimation w.r.t Fermi Constant**

$$r(x, G_F, G_F^0) = \frac{Z \to \mu\mu \,|\, G_F}{Z \to \mu\mu \,|\, G_F^0}$$

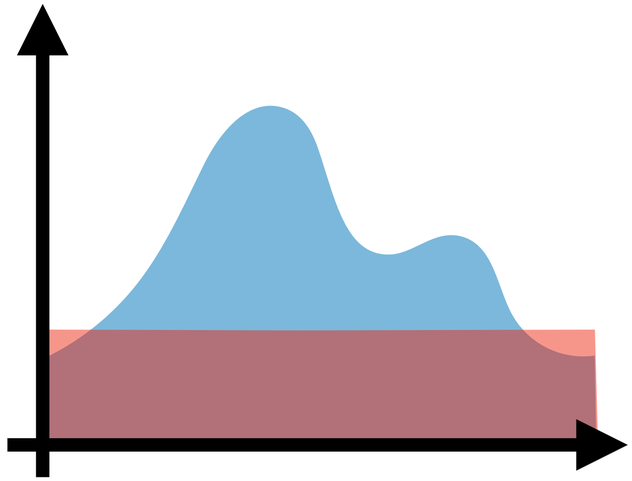




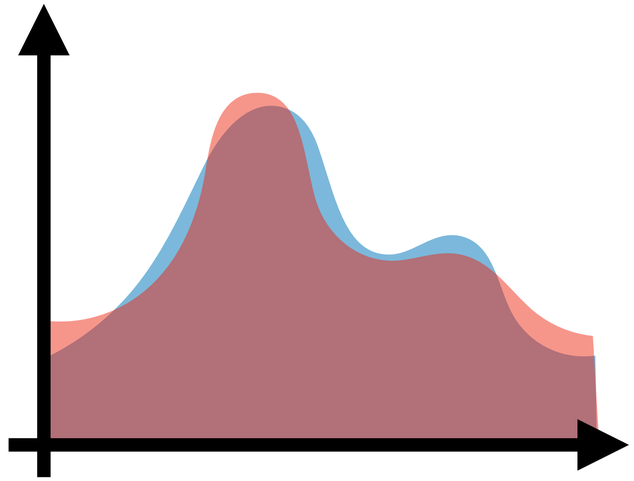
Example Use-Case II: Proposal Distribution for MC Integration

By far most important use-case for Matrix Elements:

- **calculate total cross-section $\sigma(x\,|\,\theta)$ and sample $x \sim p(x\,|\,\theta) = \mathrm{d}\sigma(x\,|\,\theta)/\sigma(\theta)$**
- **Importance Sampling: use good proposal distribution $q(x\,|\,\theta) \sim p(x\,|\,\theta)$ to reduces variance of $\sigma(\theta)$ (e.g. VEGAS)**
- **gradients of MEs can help find good proposals in a data-efficient way**



naive proposal



good proposal

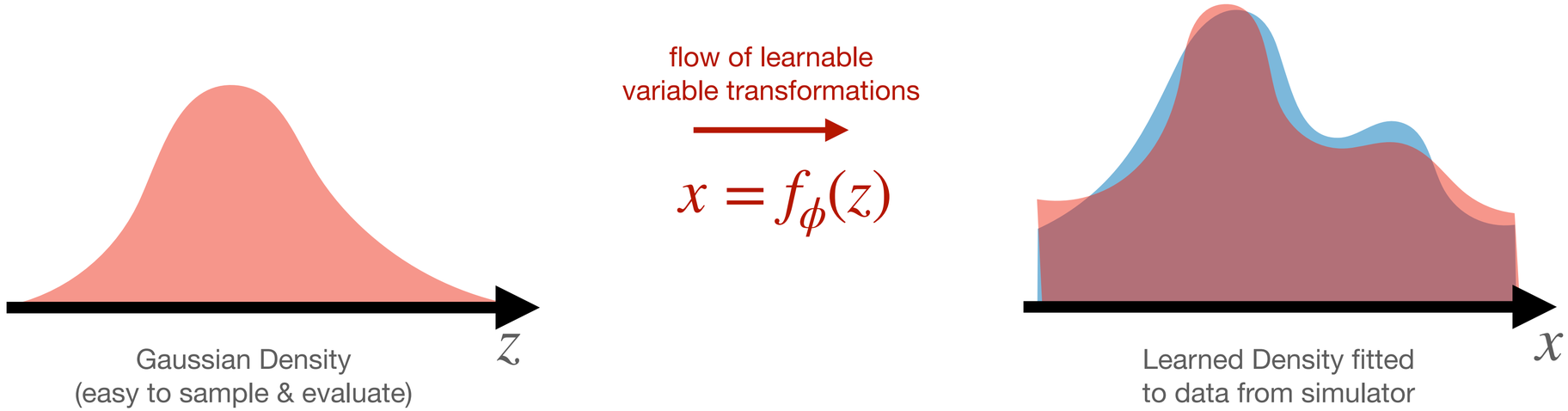
Example Use-Case II: Proposal Distribution for MC Integration

Natural Idea: use ML to find good proposal distribution.

- **ML excels solving problems approximately - just what we need**

Natural Tool: Normalizing Flows (e.g. see arXiv:2001.05486)

- **flexible density that is easy to evaluate & sample**
- **start from normal distribution and apply repeated invertible change of variables**



Evaluate: $q(x)\,\mathrm{d}x = q_{\mathcal{N}}(f^{-1}(z))\,J_f(x)\,\mathrm{d}z$

Sample: $x = f(z)$ with $z \sim q_{\mathcal{N}}(z)$

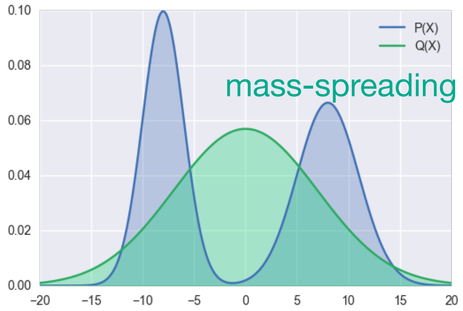
Example Use-Case II: Proposal Distribution for MC Integration

Neural Flows are trained to minimize empirical Kullback-Leibler (KL) distance

- **asymmetry of KL yields two complementary approaches**

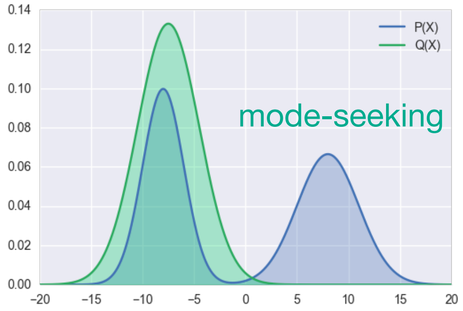
"forward KL"
(expensive samples)

$$L_{\mathrm{fKL}}(\phi) = \mathrm{KL}(p\,||\,q_\phi) \rightarrow \sum_{x_i \sim p} \log p(x_i)/q_\phi(x_i)$$



"reverse KL"
(cheap samples)

$$L_{rKL}(\phi) = \mathrm{KL}(q_\phi\,||\,p) \rightarrow \sum_{x_i \sim q_\phi} \log q_\phi(x_i)/p(x_i)$$



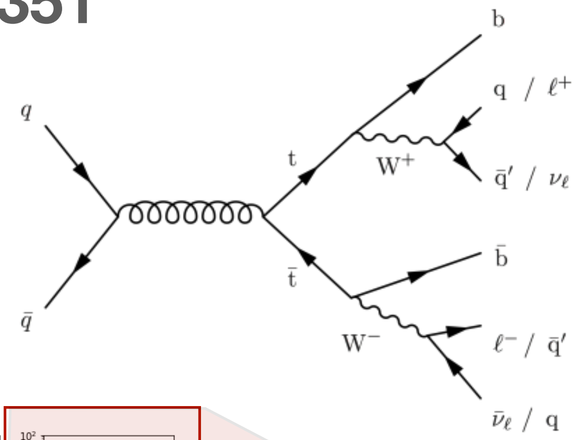
Training on combination: $L = L_{fKL} + \omega L_{rKL}$ captures mix of both behaviors

- **but reverse KL needs gradients of unnormalized values of $p(x)$**

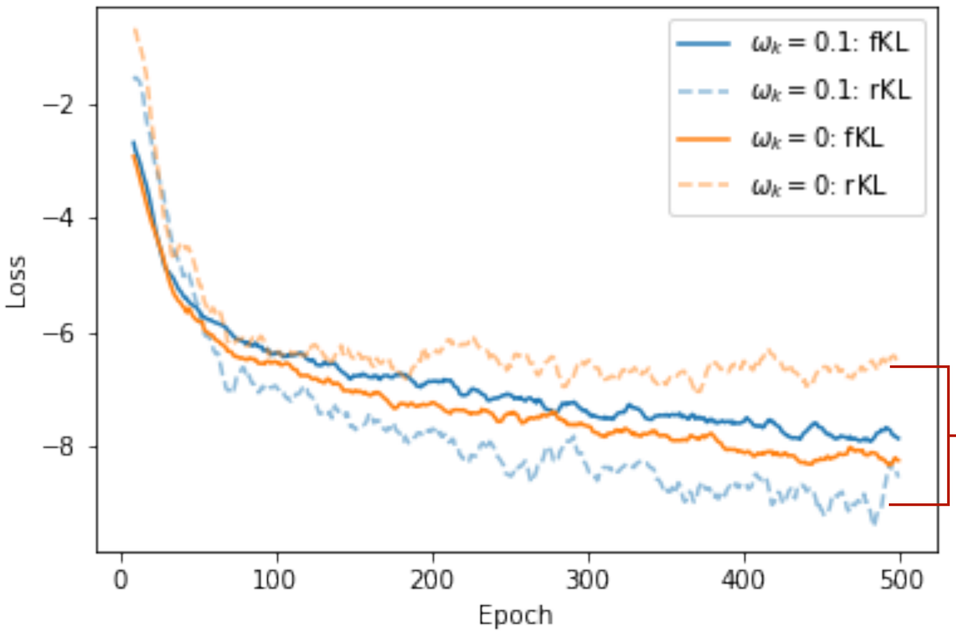
Example Use-Case II: Proposal Distribution for MC Integration

Implementing "Smooth Normalizing Flows" from arXiv:2110.00351

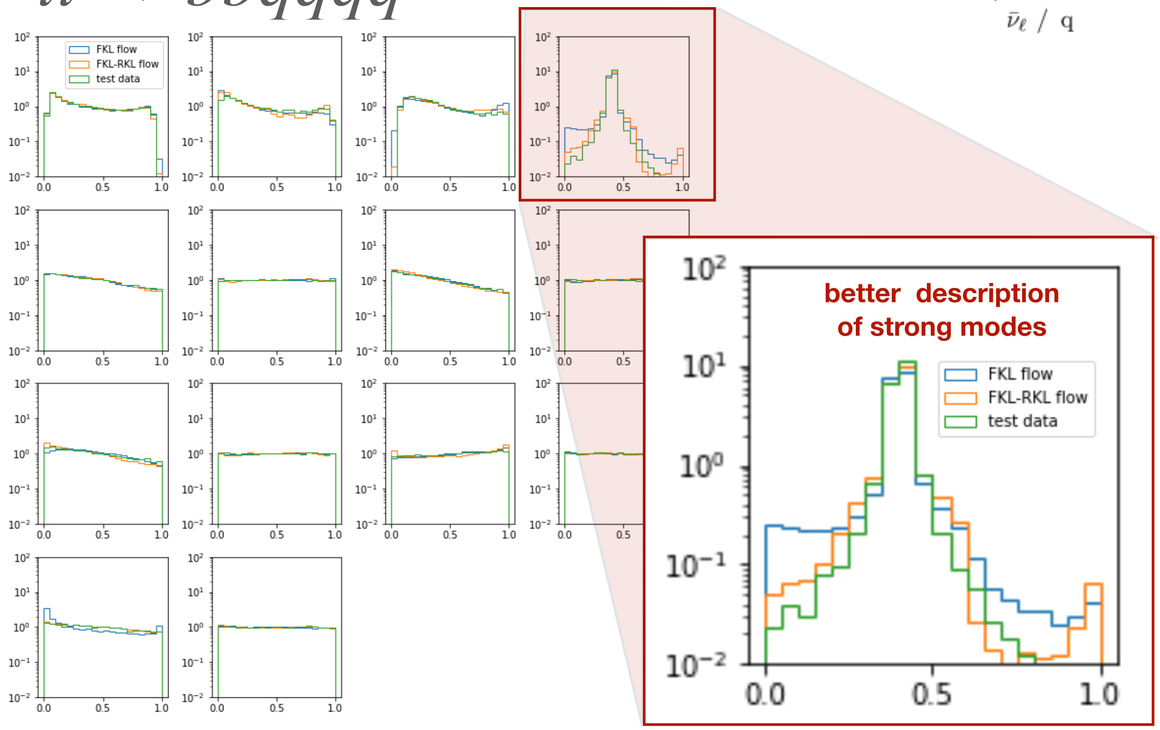
- method for neural flows on bounded manifolds
- trains on mixture of KL distances
- data-efficient: flow samples for rKL are cheap!



14-D Example: $qq \to tt \to bbqqqq$



improvement in flow description of ME



Summary & Outlook

Machine Learning renewed focus on algorithmic differentiation of scientific code.

Matrix Element Code (with flexible code generation) a prime target in HEP

- **Gradients (both wrt. theory and phase-space) useful for many applications with and without ML**
- **MadJax provides differentiable Matrix-Elements by using autodiff with JAX**
- **enables a number of previously inaccessible use-cases**

If you have use for ME gradients, please let us know!

- **always happy to collaborate**