Exploring Phase Space with Neural Importance Sampling [2001.05478]

Enrico Bothmann Timo Janßen Max Knobbe Tobias Schmale
Steffen Schumann

Institut für Theoretische Physik, Georg-August-Universität Göttingen

ML4Jets2021

Outline

- Monte Carlo Event Generators
- Neural Importance Sampling (Müller et al. 2018)
- Results

Monte Carlo Event Generators

Goals

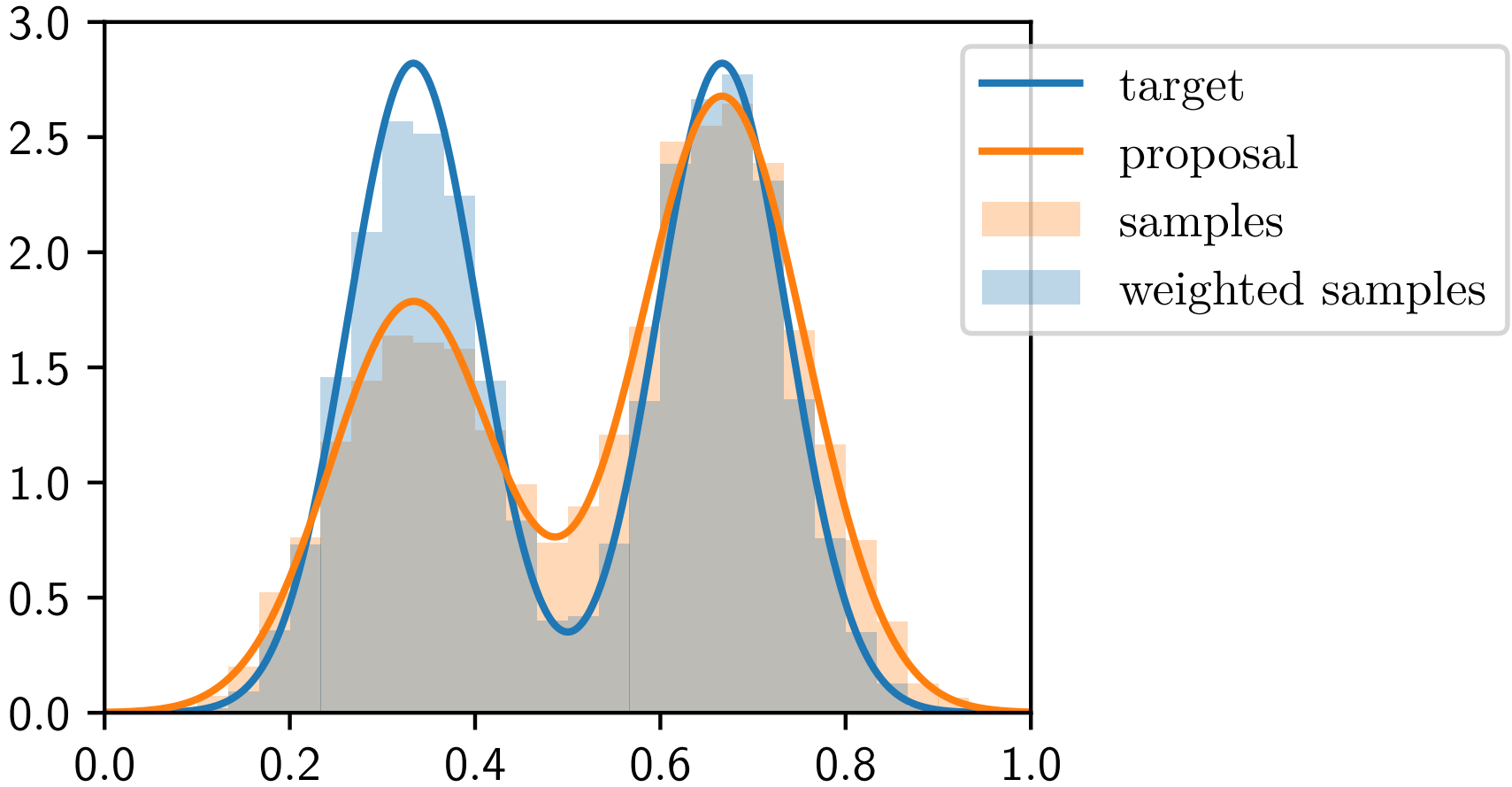
1. calculate total cross section σ
2. generate events

Procedure

1. generate valid PS point
2. evaluate amplitude $\rightarrow$ weight
3. accept or reject point
4. parton shower, hadronisation, ...

How to generate proposal points

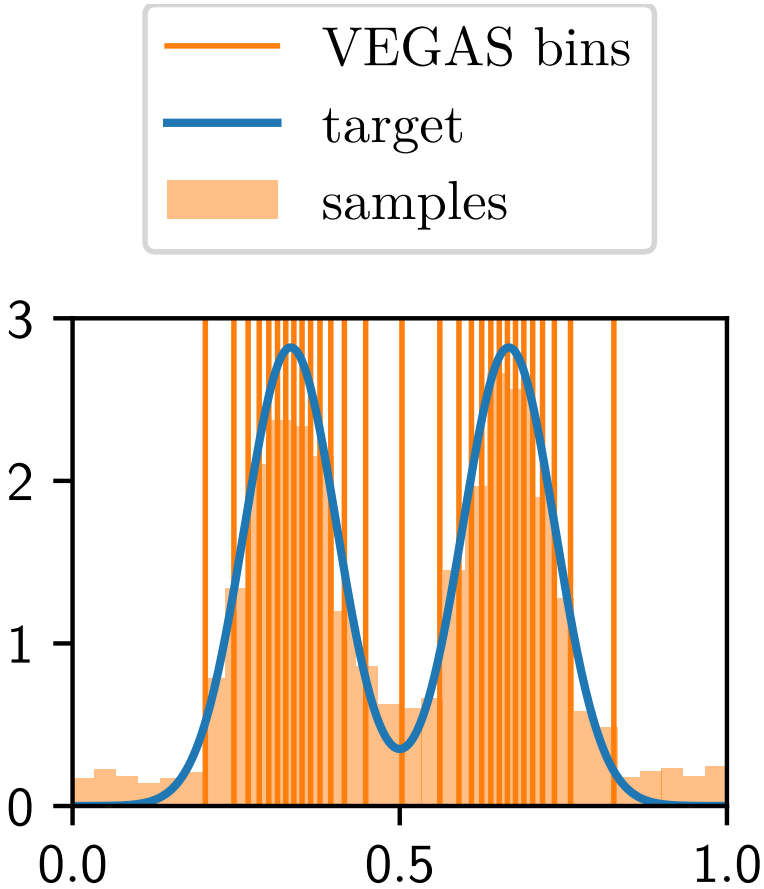
multi-channel importance sampling:



mixture weights can be optimised automatically (Kleiss and Pittau 1994)

How to generate proposal points

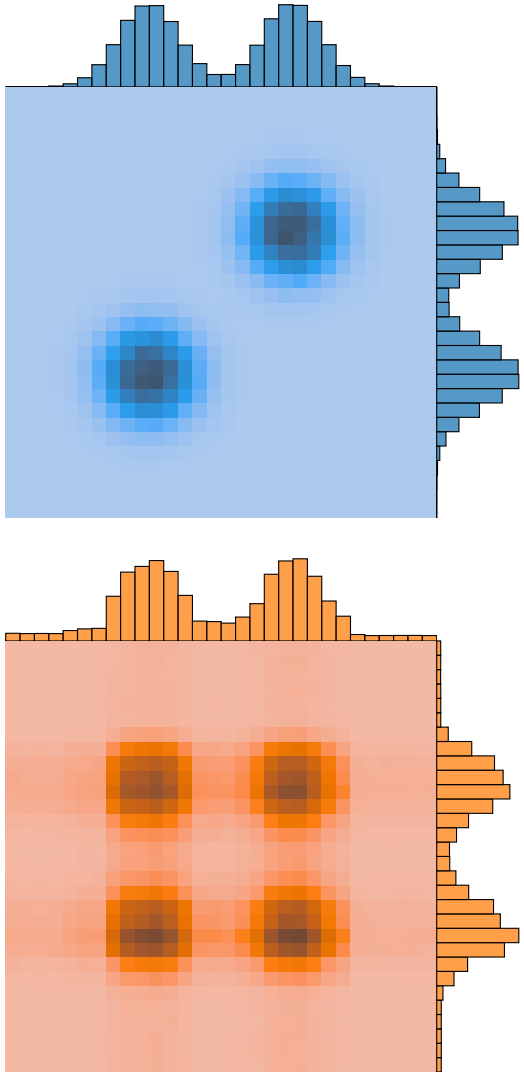
adaptive importance sampling: VEGAS (Peter Lepage 1978)



- adapt piecewise constant fn to marginals of target
- partition each dim into bins & adapt bin widths
  - many narrow bins where target is peaked
  - few wide bins where target is flat

How to generate proposal points

adaptive importance sampling: VEGAS (Peter Lepage 1978)

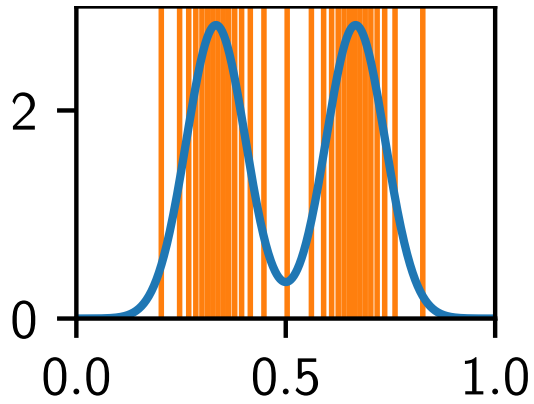


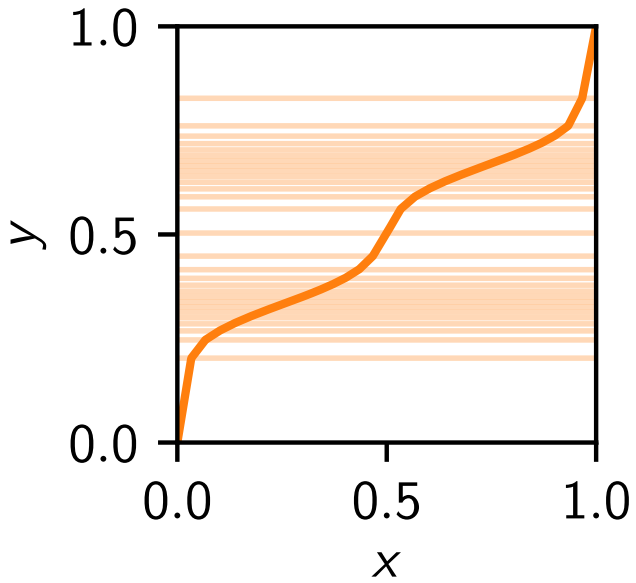
Why not VEGAS for everything?

- convergence slow for highly peaked and/or high-dimensional targets
- when peaks aren't aligned with coord. axes, VEGAS learns "phantom peaks"

How to generate proposal points

adaptive importance sampling: VEGAS (Peter Lepage 1978)





VEGAS as a mapping

- VEGAS can be regarded as a bijective mapping between unit hypercubes.
- can be used to improve another proposal
- only has to learn the mismatch

State of the Art in SHERPA

- construct multi-channel distribution based on Feynman diagrams
  - typical building blocks: Breit-Wigner, QCD antennae, (an)isotropic decays, ...
- dress each channel with a VEGAS mapping
- optimise mixture weights and VEGAS mappings in an initial optimisation phase
- This is all fully automated!

Outline

- Monte Carlo Event Generators
- Neural Importance Sampling (Müller et al. 2018)
- Results

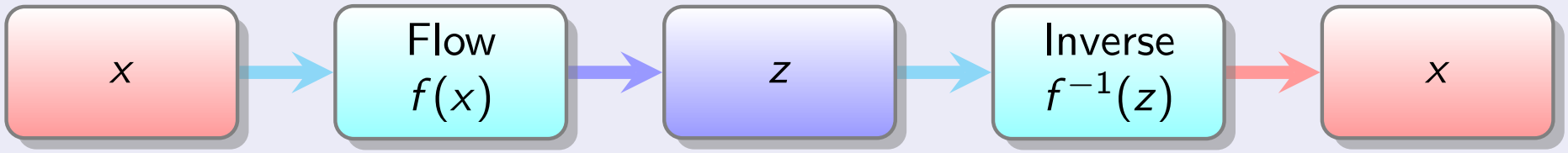
Can we do better using Deep Learning?

Our requirements:

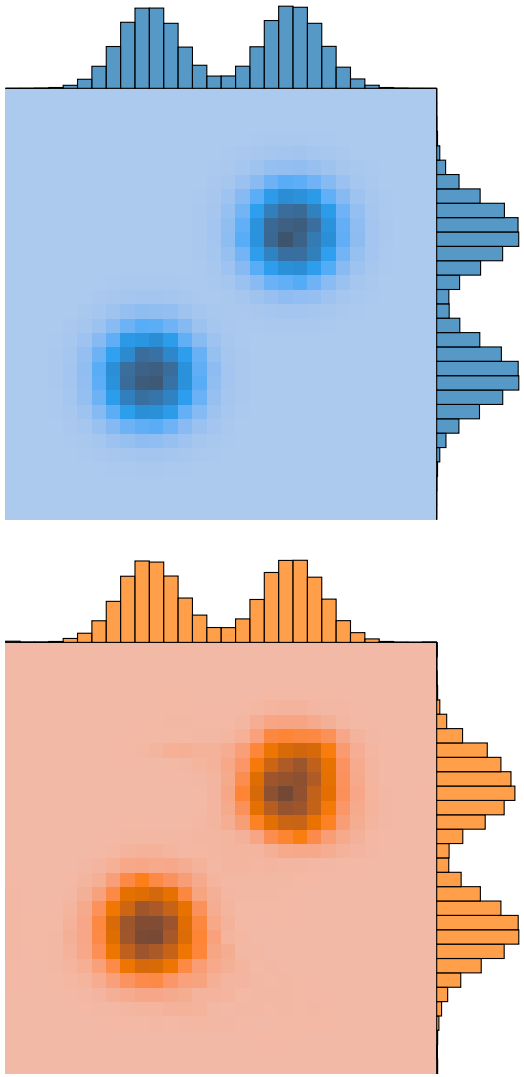
- full phase space coverage
- convergence to the target distribution
- general method, lending itself to automation
- uncorrelated events

Our tool of choice: Normalizing Flows

- provide bijective mapping with analytic inverse
- use NF as a direct replacement for VEGAS



Piecewise Rational-Quadratic Coupling Layers (Durkan et al. 2019)



- variables live in unit hypercube: $h : [0,1]^d \to [0,1]^d$
- interpret components C_i of Coupling Transform as CDF
- model CDF with piecewise rational-quadratic splines using K knots
- the parameters are determined by a Neural Network

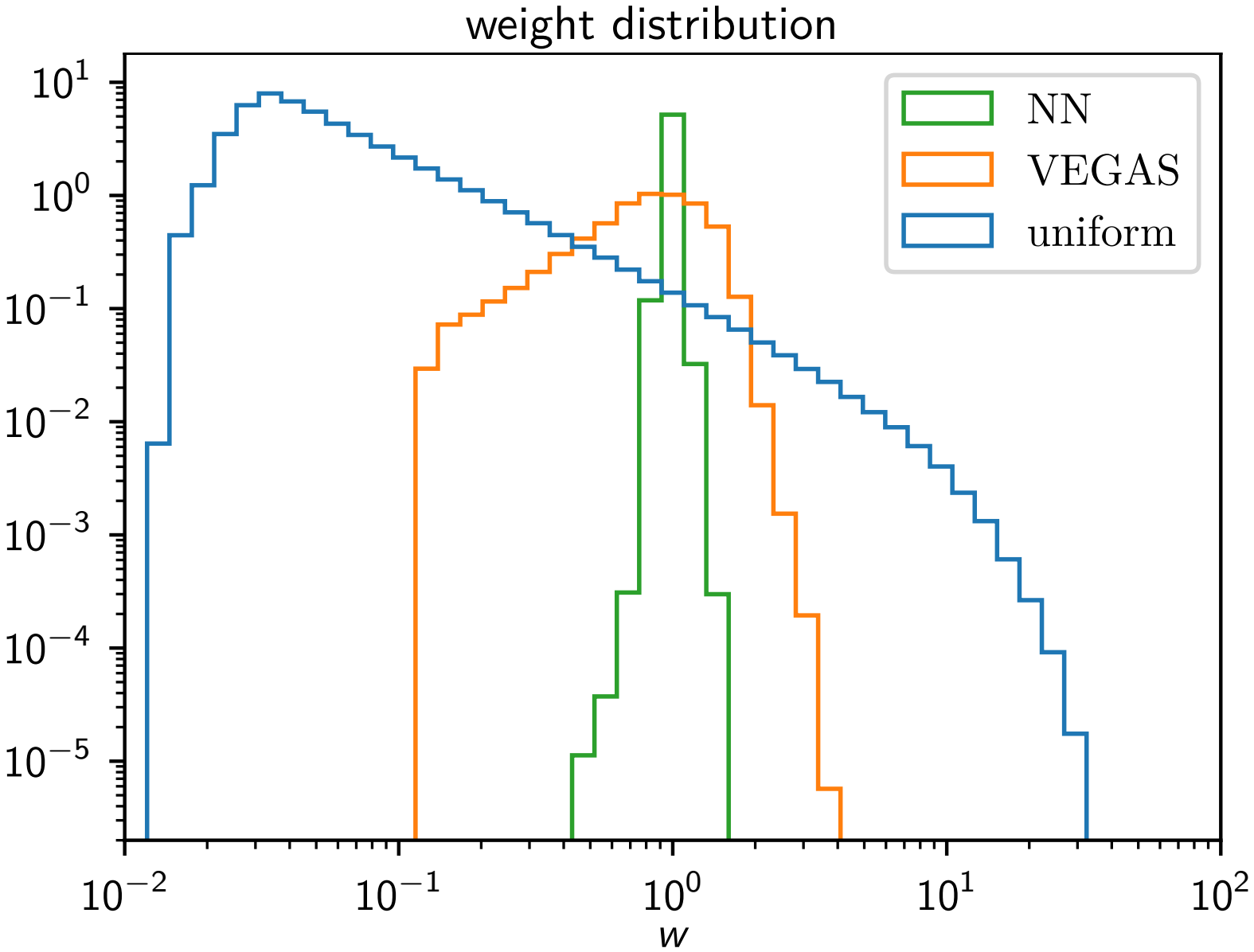
Outline

- Monte Carlo Event Generators
- Neural Importance Sampling (Müller et al. 2018)
- Results

Gluon Scattering: $gg \to 3g$

- $\sqrt{s} = 1\,\text{TeV}$
- multi channel: 2 independent channels
- use HAAG phase space mapping (van Hameren and Papadopoulos 2002)
- 5 phase space dimensions
- cuts: $p_\perp > 30\,\text{GeV}$ and $m_{ij} > 30\,\text{GeV}$

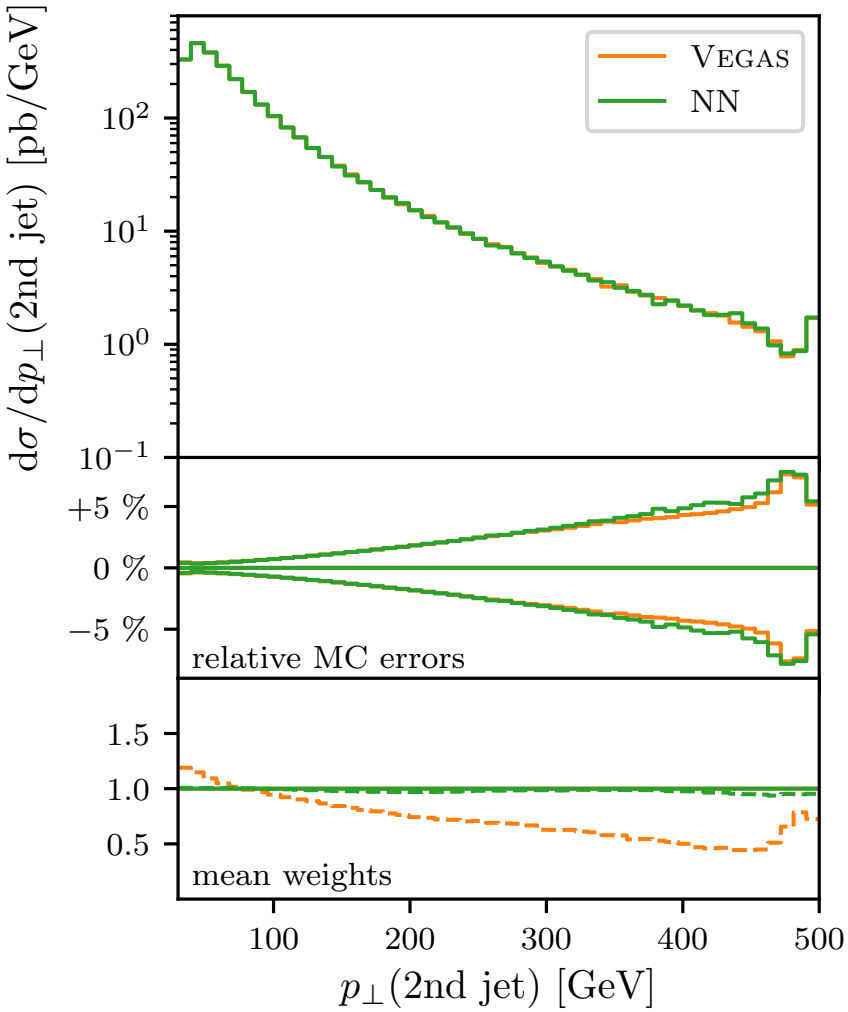
Gluon Scattering: $gg \to 3g$

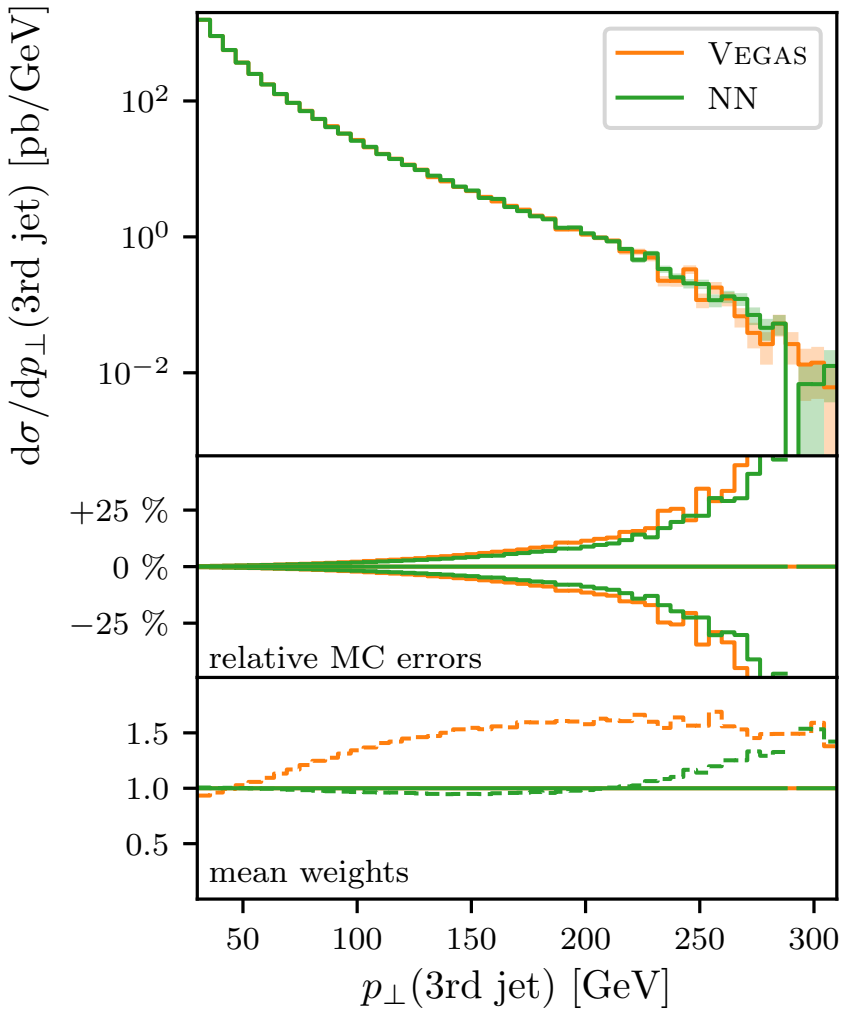


Gluon Scattering: $gg \to 3g$

Sample	ϵ_{uw}	E_N [pb]
Uniform	3.0 %	24806(55)
VEGAS	27.7 %	24813(23)
NN	64.3 %	24847(21)

Gluon Scattering: $gg \to 3g$

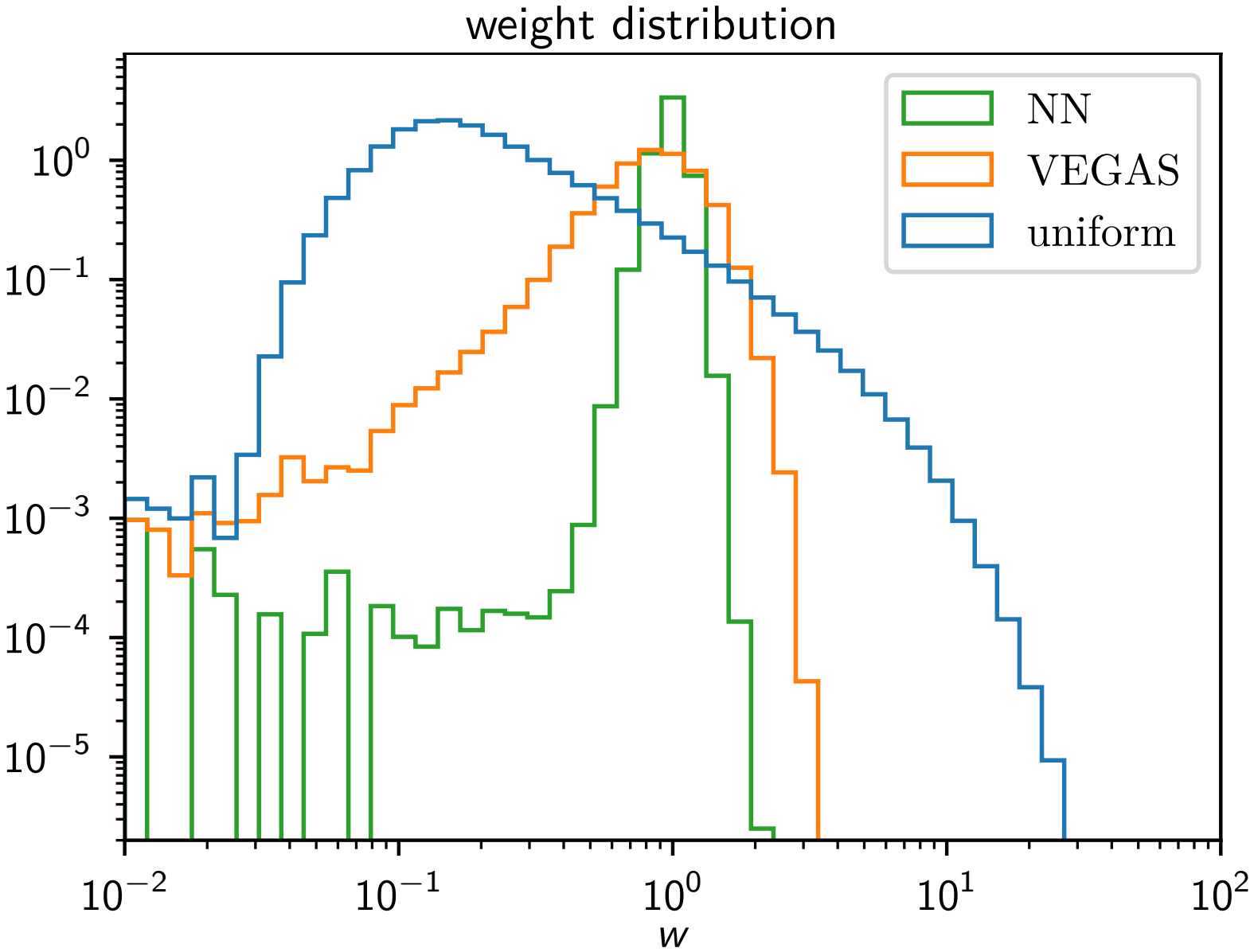




Gluon Scattering: $gg \to 4g$

- $\sqrt{s} = 1\,\text{TeV}$
- multi channel: 3 independent channels
- use HAAG phase space mapping (van Hameren, Papadopoulos (2002))
- 8 phase space dimensions
- cuts: $p_\perp > 30\,\text{GeV}$ and $m_{ij} > 30\,\text{GeV}$

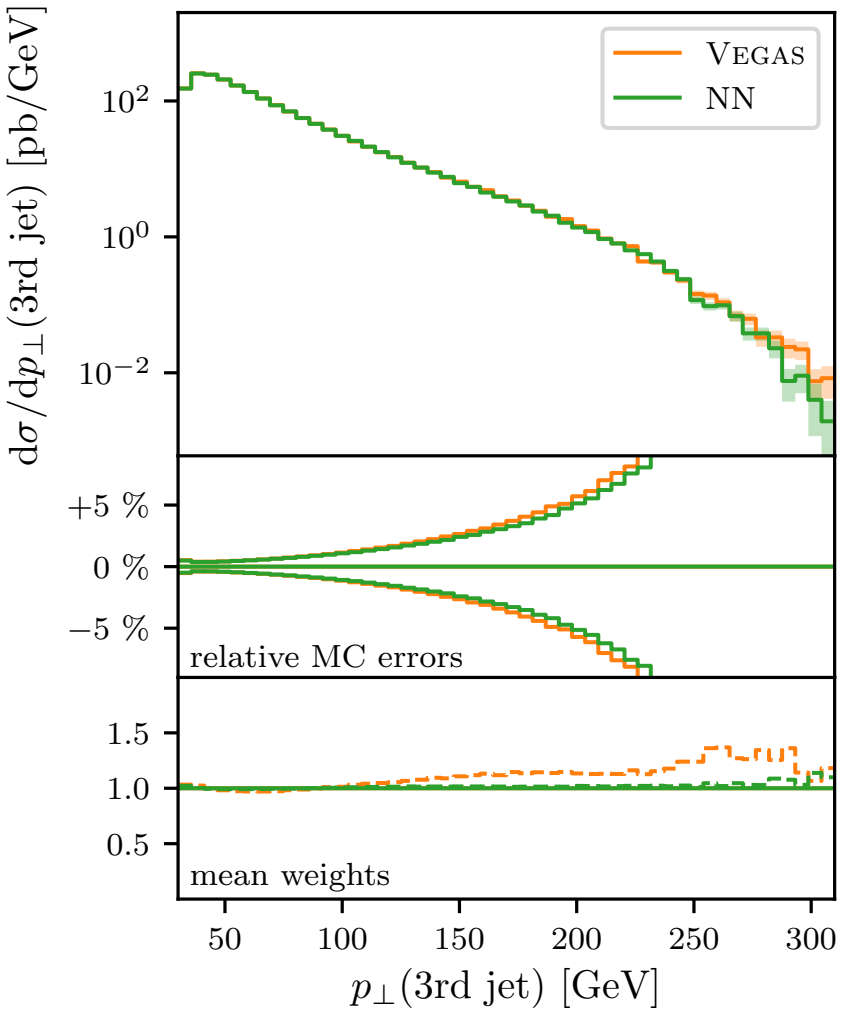
Gluon Scattering: $gg \to 4g$

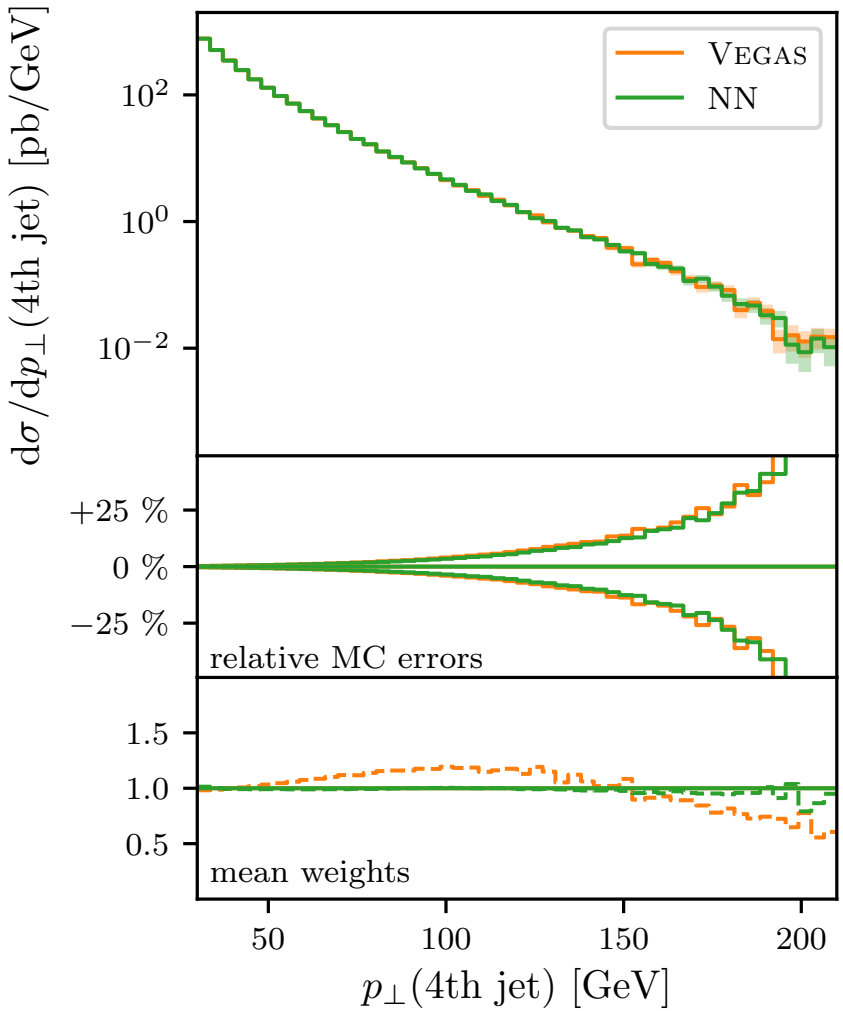


Gluon Scattering: $gg \to 4g$

Sample	ϵ_{uw}	E_N [pb]
Uniform	2.7 %	9869(20)
VEGAS	31.8 %	9868(10)
NN	~~33.6 %~~ 48.9 %	9859(10)

Gluon Scattering: $gg \to 4g$





Train directly on Sample

- idea: learn from a weighted training sample
- similar to (Stienen and Verheyen 2020)
- use RAMBO on diet (Plätzer 2013) to map physical phase space to unit hypercube

pros:

- single channel $\rightarrow$ only one NF
- can be used easily with any existing generator

cons:

- lose some prior knowledge
- need to prevent overtraining

Train directly on Sample

Results:

3 jets

Sample	ϵ_{uw}
Uniform multi-channel	3 %
VEGAS multi-channel	28 %
NN multi-channel	64 %
NN single channel	58 %

4 jets

Sample	ϵ_{uw}
Uniform multi-channel	3 %
VEGAS multi-channel	32 %
NN multi-channel	49 %
NN single channel	11 %

Conclusions

- proof of principle: event generation with normalizing flows
- pros: phase-space coverage guaranteed, cheap weight determination
- as drop-in replacement for VEGAS in multi-channel:
  - improved unw. eff. for up to $d = 8$
- as single channel trained on sample:
  - improved unw. eff. for up to $d = 5$
  - limited performance for $d = 8$
- other study (Gao et al. 2020) comes to similar conclusions:
  - factor 2-3 in unw. eff. for simple processes
  - similar to multi-channel+VEGAS for high multiplicities

Outlook

- GPU matrix elements $\rightarrow$ cf. (Bothmann et al. 2021)
  talk: Joshua Isaacson Wed 7/7/21 4:40 PM
- reduce training cost
- alternatives to piecewise rational-quadratic coupling layers
- improve scaling behaviour of single channel method
- there are also other parts in MC Event Generators that can be optimised

References I

[1] E. Bothmann, W. Giele, S. Hoeche, J. Isaacson, and M. Knobbe, "Many-gluon tree amplitudes on modern GPUs: A case study for novel event generators", arXiv e-prints, arXiv:2106.06507, arXiv:2106.06507 (2021).

[2] L. Dinh, D. Krueger, and Y. Bengio, "NICE: non-linear independent components estimation", in 3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, workshop track proceedings, edited by Y. Bengio and Y. LeCun (2015).

[3] C. Durkan, A. Bekasov, I. Murray, and G. Papamakarios, "Neural spline flows", in Advances in neural information processing systems, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (2019).

[4] C. Gao, S. Höche, J. Isaacson, C. Krause, and H. Schulz, "Event generation with normalizing flows", **101**, 076002, 076002 (2020).

References II

[5] R. Kleiss, W. J. Stirling, and S. D. Ellis, "A new monte carlo treatment of multiparticle phase space at high-energies", Comput. Phys. Commun. **40**, 359 (1986).

[6] R. Kleiss and R. Pittau, "Weight optimization in multichannel monte carlo", Comput. Phys. Commun. **83**, 141–146 (1994).

[7] T. Müller, B. McWilliams, F. Rousselle, M. Gross, and J. Novák, "Neural importance sampling", CoRR **abs/1808.03856** (2018).

[8] G. Peter Lepage, "A new algorithm for adaptive multidimensional integration", Journal of Computational Physics **27**, 192–203 (1978).

[9] S. Plätzer, "Rambo on diet", (2013).

[10] B. Stienen and R. Verheyen, "Phase space sampling and inference from weighted events with autoregressive flows", arXiv e-prints, arXiv:2011.13445, arXiv:2011.13445 (2020).

References III

[11] A. van Hameren and C. G. Papadopoulos, "A hierarchical phase space generator for QCD antenna structures", European Physical Journal C **25**, 563–574 (2002).

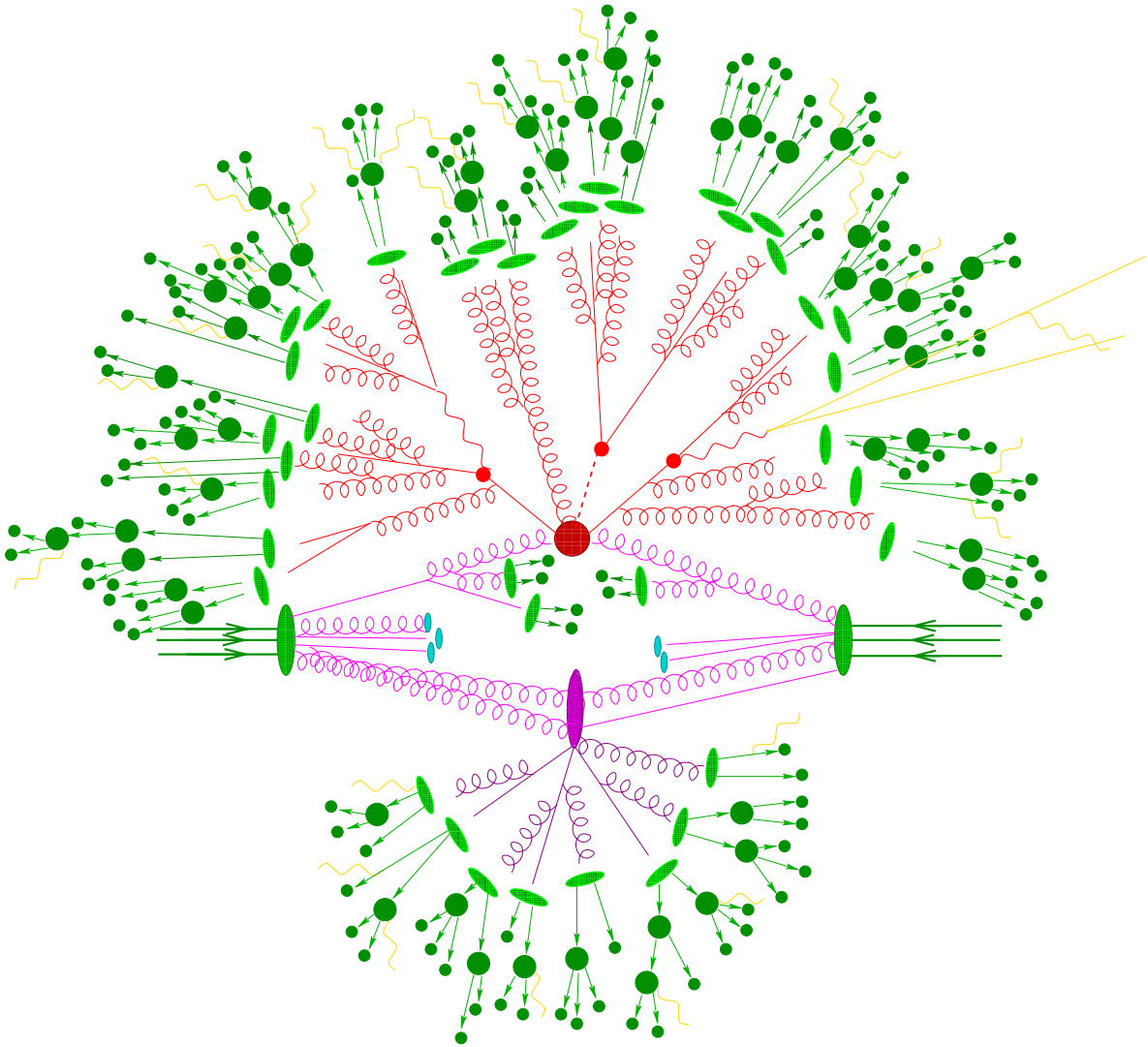
Backup Slides

Motivation

- LHC HL upgrade
  $\rightarrow$ more statistics
  $\rightarrow$ access to higher multiplicities
- Event Generators reliable and fully automated for arbitrary processes
- but high multiplicities suffer from low efficiencies
  $\rightarrow$ CPU resource need is high
- expected discrepancy between resource needs and budget
- more efficient sampling can be part of the solution

Monte Carlo Event Generators

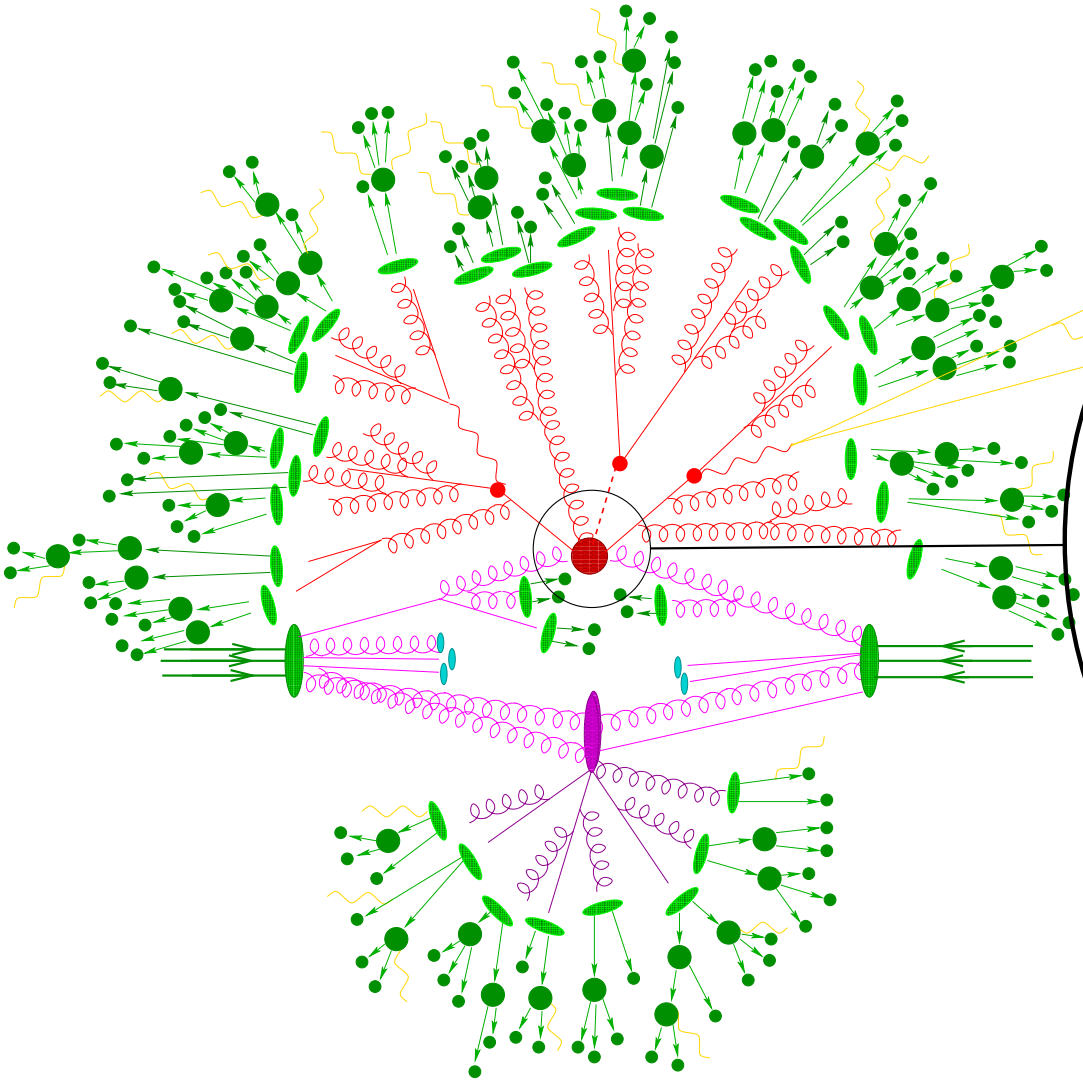
An event:

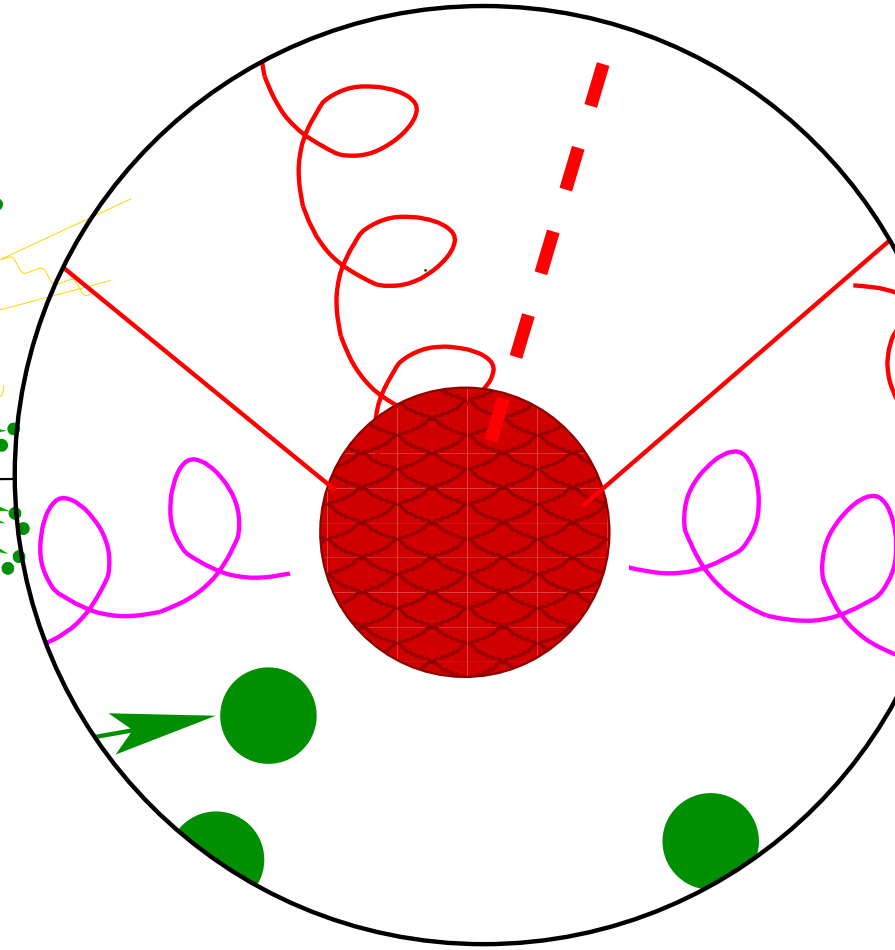


Monte Carlo Event Generators

An event:

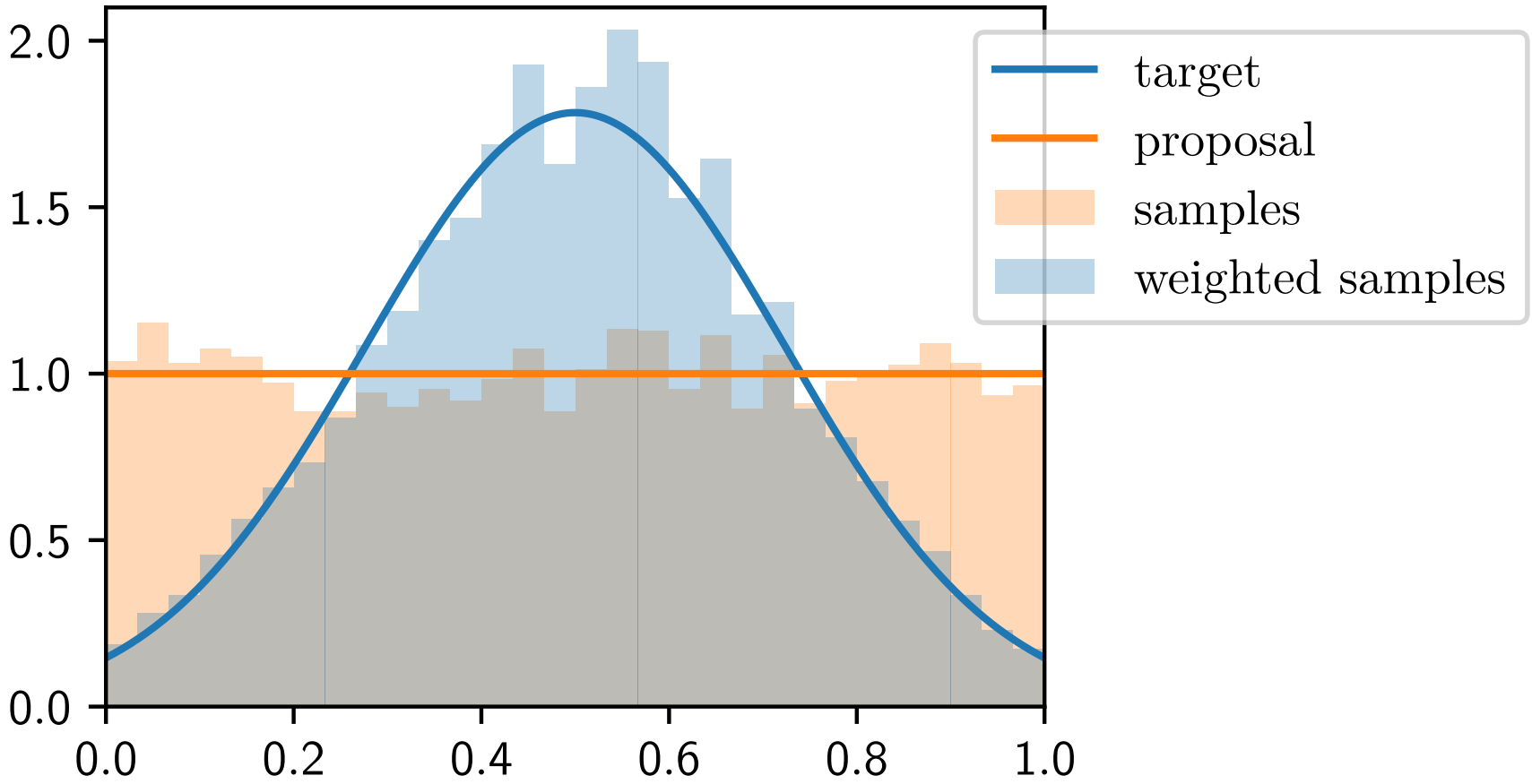
Parton-level:





How to generate proposal points

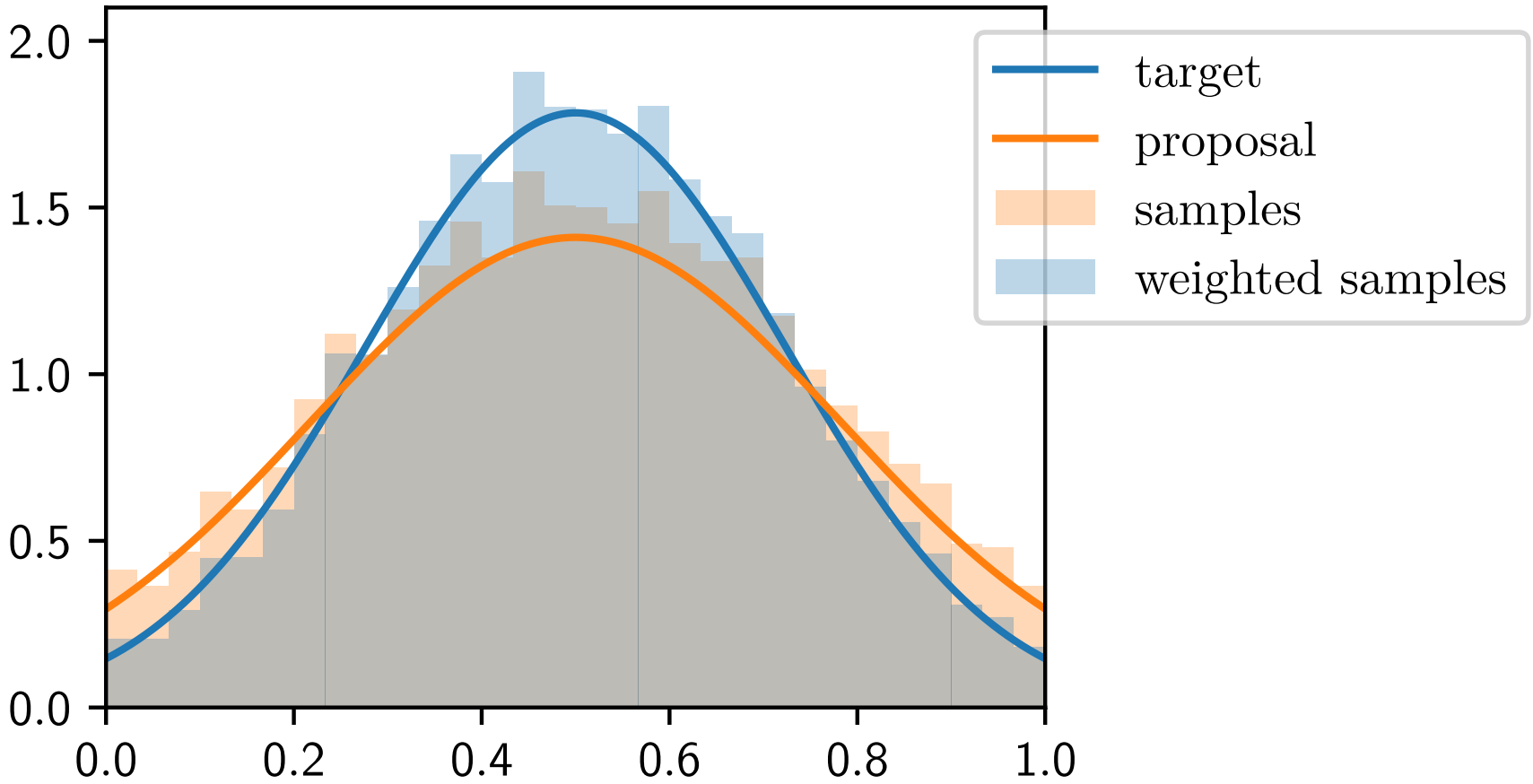
classic Monte Carlo: sample uniformly



HEP equivalent: RAMBO (Kleiss, Stirling, et al. 1986)

How to generate proposal points

importance sampling:



HEP example: Breit-Wigner distribution for resonances

Coupling Layers (Dinh et al. 2015)

Consider a mapping

$$h : X \to Y,$$
$$x \sim p_X(x) \mapsto y \in Y$$

The PDF of y is given by

$$p_Y(y) = p_X(x) \left| \det \left(\frac{\partial h(x)}{\partial x^T} \right) \right|^{-1}$$

Numerical computation of arbitrary determinant costs $\mathcal{O}(d^3)$!

Coupling Layers (Dinh et al. 2015)

Idea: split input into two partitions $x = (x^A, x^B)$ and map to

$$y^A = x^A,$$
$$y^B = C(x^B; m(x^A))$$

The Coupling Transform C is an invertible and separable map.

Jacobian

$$\frac{\partial y(x)}{\partial x^T} = \begin{pmatrix} I_n & 0 \\ \frac{\partial C(x^B; m(x^A))}{\partial (x^A)^T} & \frac{\partial C(x^B; m(x^A))}{\partial (x^B)^T} \end{pmatrix} \quad \leftarrow \text{Triangular!}$$

$\rightarrow$ Determinant costs $\mathcal{O}(d)$

- $\rightarrow$ Jac does not involve $\frac{\partial m(x^A)}{\partial x^A}$
- $\rightarrow$ m can be arbitrarily complex
- $\rightarrow$ use a Neural Network

Coupling Layers (Dinh et al. 2015)

A single Coupling Layer transforms only part of the input

$\rightarrow$ Use a layered mapping: $h = h_L \circ \cdots \circ h_2 \circ h_1$

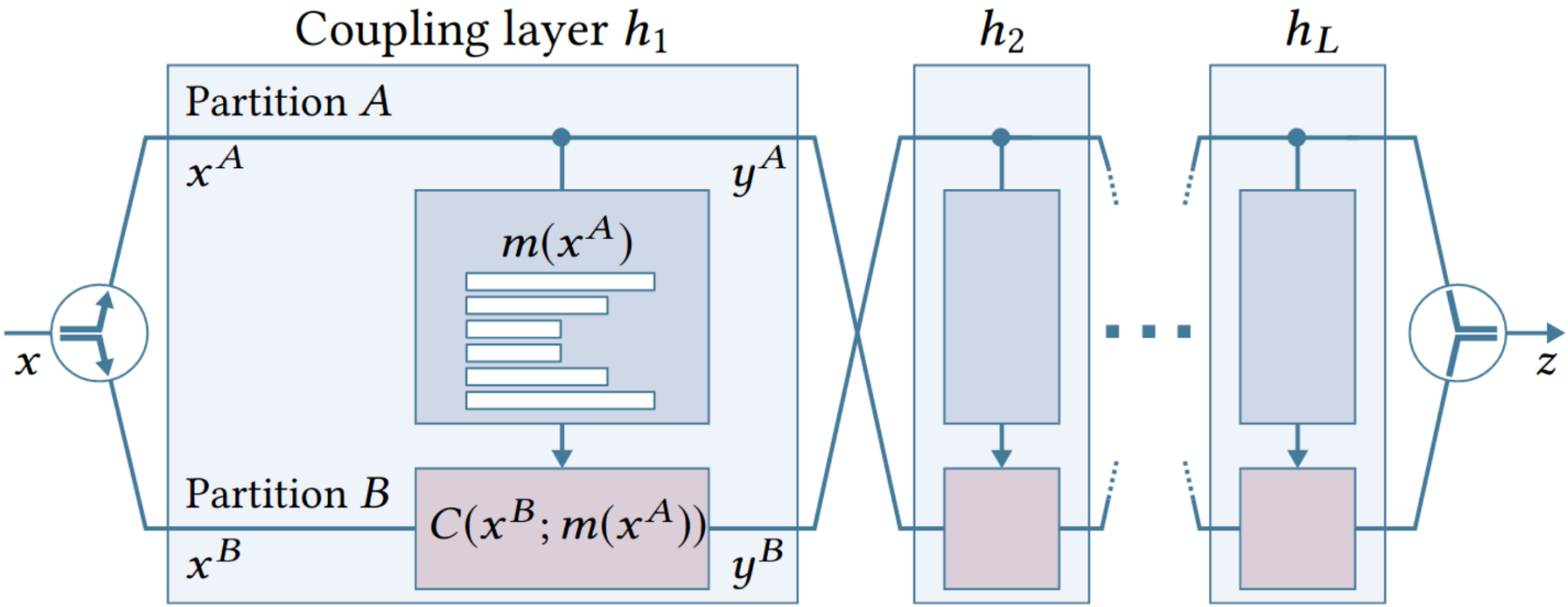
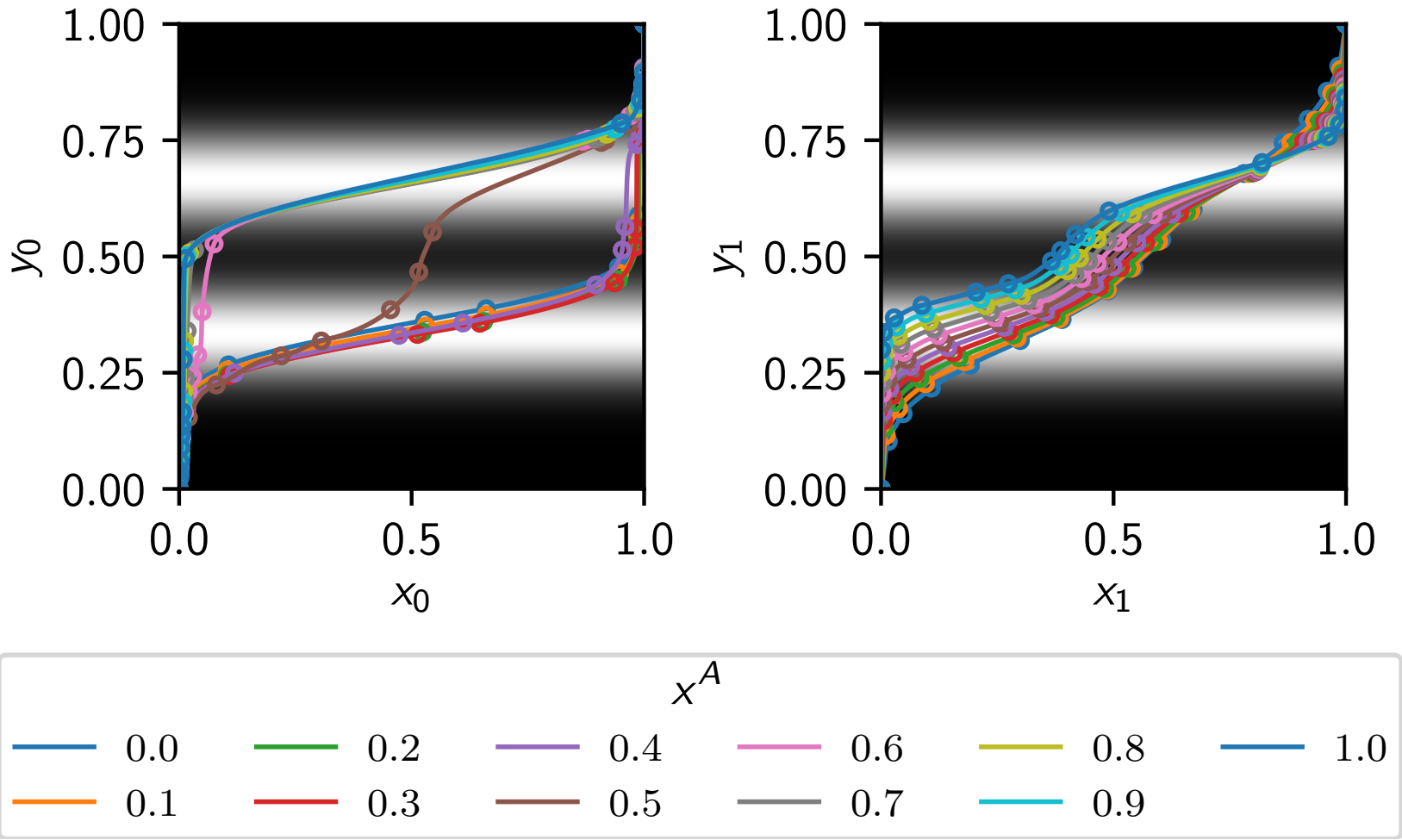


image source: Müller et al. 2018, arXiv:1808.03856

Piecewise Rational-Quadratic Coupling Layers (Durkan et al. 2019)



Importance Sampling with Coupling Layers

To generate one event:

1. draw $x \in [0,1]^d$ from a uniform distribution
2. choose a channel c at random
3. map $x \mapsto y \in [0,1]^d$ using several Coupling Layers
4. use channel mapping $g_c(y)$ to get a set of momenta p
5. weight:

$$w = \left|\det\left(\frac{\partial y(x)}{\partial x^T}\right)\right| \left|\det\left(\frac{\partial p(y)}{\partial y^T}\right)\right| f(p)$$

Training

Minimise mean squared error distance in minibatches:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(f(p_i) - g(p_i))^2$$

Use gradient descent to optimise the weights of the Neural Networks

Controlling the Cut Efficiency

- using MSE loss we can control the cut efficiency by normalizing the target
- allow NF to place some probability in the cut region
  $\rightarrow$ better approximate target in physical phase space
- tradeoff between unweighting efficiency and cut efficiency