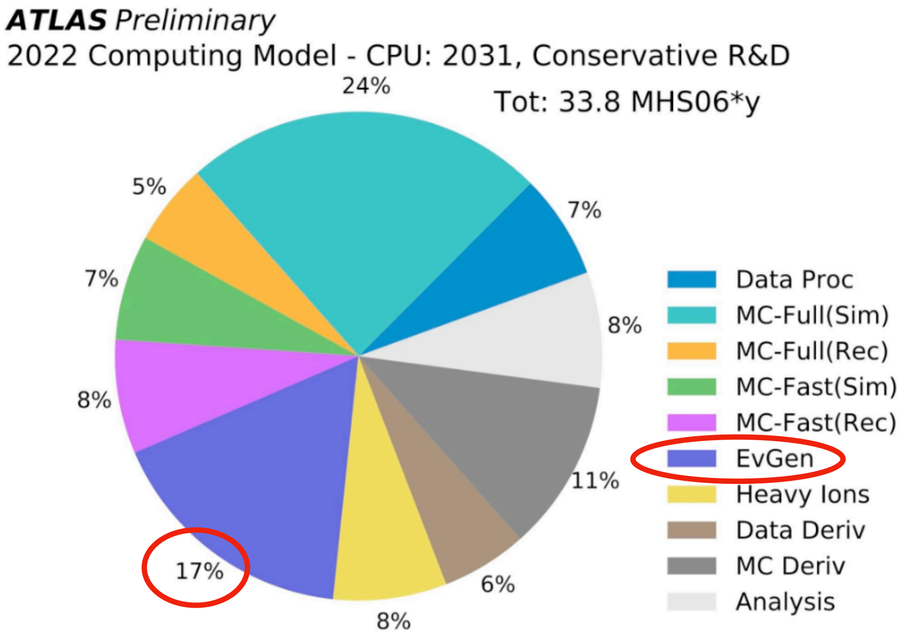
MadNIS and the road to MadGraph 7

Theo Heimel
March 2025

CP3, UCLouvain

[2212.06172] TH, Winterhalder, Butter, Isaacson, Krause, Maltoni, Mattelaer, Plehn
[2311.01548] TH, Huetsch, Maltoni, Mattelaer, Plehn, Winterhalder
[2408.01486] TH, Mattelaer, Plehn, Winterhalder

Introduction

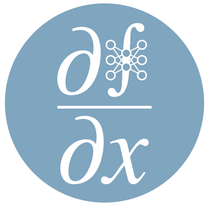


[CERN-LHCC-2022-005]

- Event generation **bottleneck: hard process**
- Orthogonal approaches
  → hardware acceleration (GPU, SIMD, …)
  Sherpa/Pepper: 2302.10449, 2311.06198
  MadGraph: 2303.18244, 2312.02898, 2312.07440
  → fast ML surrogates [Nina's and Tim's talks]
  Sherpa: 2109.11964, 2301.13562
  MadGraph: 2412.12069
  → **ML-based sampling: MadNIS**
  Sherpa: 2001.05486, 2001.05478, 2001.10028,
  MadGraph: 2212.06172, 2311.01548, 2408.01486
- neural importance sampling
  → phase space weight available
  → **exact theory prediction**
  → better model = faster sampling

Outline

Introduction to MadNIS



Differentiability and phase space

The road to MadGraph 7

Event generation in MadGraph

Calculate (differential) cross sections

$$\mathrm{d}\sigma = \frac{1}{\text{flux}} \mathrm{d}x_a \mathrm{d}x_b \, f(x_a) f(x_b) \, \mathrm{d}\Phi_n \left\langle \, |M_{\lambda,c,\ldots}(p_a, p_b \mid p_1, \ldots, p_n)|^2 \, \right\rangle$$

Sum over channels

MadGraph: build channels from Feynman diagrams

Integrand

MadGraph: $\mathrm{d}\sigma/\mathrm{d}x$

$$I = \sum_i \left\langle \alpha_i(x) \frac{f(x)}{p_i(x)} \right\rangle_{x \sim p_i(x)}$$

Channel weights

MadGraph: $\alpha_i^{\mathrm{MG}}(x) \sim |M_i|^2$

Channel mappings

MadGraph: use amplitude structure, ...

Analytic mappings + refine with **VEGAS** (factorized, histogram based importance sampling)

Event generation in MadNIS

Calculate (differential) cross sections

$$\mathrm{d}\sigma = \frac{1}{\text{flux}} \mathrm{d}x_a \mathrm{d}x_b \, f(x_a) f(x_b) \, \mathrm{d}\Phi_n \left\langle \, |M_{\lambda,c,\ldots}(p_a, p_b \mid p_1, \ldots, p_n)|^2 \, \right\rangle$$

Sum over channels

MadGraph: build channels from Feynman diagrams

Integrand

MadGraph: $\mathrm{d}\sigma/\mathrm{d}x$

$$I = \sum_i \left\langle \alpha_i(x) \frac{f(x)}{p_i^\omega(x)} \right\rangle_{x \sim p_i^\omega(x)}$$

Channel weights

MadGraph: $\alpha_i^{\mathrm{MG}}(x) \sim |M_i|^2$

Learned channel mappings

MadGraph: use amplitude structure, ...

Analytic mappings + ~~refine with VEGAS~~

refine with Normalizing Flow

Event generation in MadNIS

Calculate (differential) cross sections

$$\mathrm{d}\sigma = \frac{1}{\text{flux}} \mathrm{d}x_a \mathrm{d}x_b \, f(x_a) f(x_b) \, \mathrm{d}\Phi_n \left\langle \, |M_{\lambda,c,\ldots}(p_a, p_b \mid p_1, \ldots, p_n)|^2 \, \right\rangle$$

Sum over channels

MadGraph: build channels from Feynman diagrams

Integrand

MadGraph: $\mathrm{d}\sigma/\mathrm{d}x$

$$I = \sum_i \left\langle \alpha_i^\xi(x) \frac{f(x)}{p_i^\omega(x)} \right\rangle_{x \sim p_i^\omega(x)}$$

Learned channel weights

MadGraph: $\alpha_i^{\mathrm{MG}}(x) \sim |M_i|^2$

$$\alpha_i(x) \to \alpha_i^\xi(x) = \alpha_i^{\mathrm{MG}}(x) \cdot K_i^\xi(x)$$

parametrize with NN

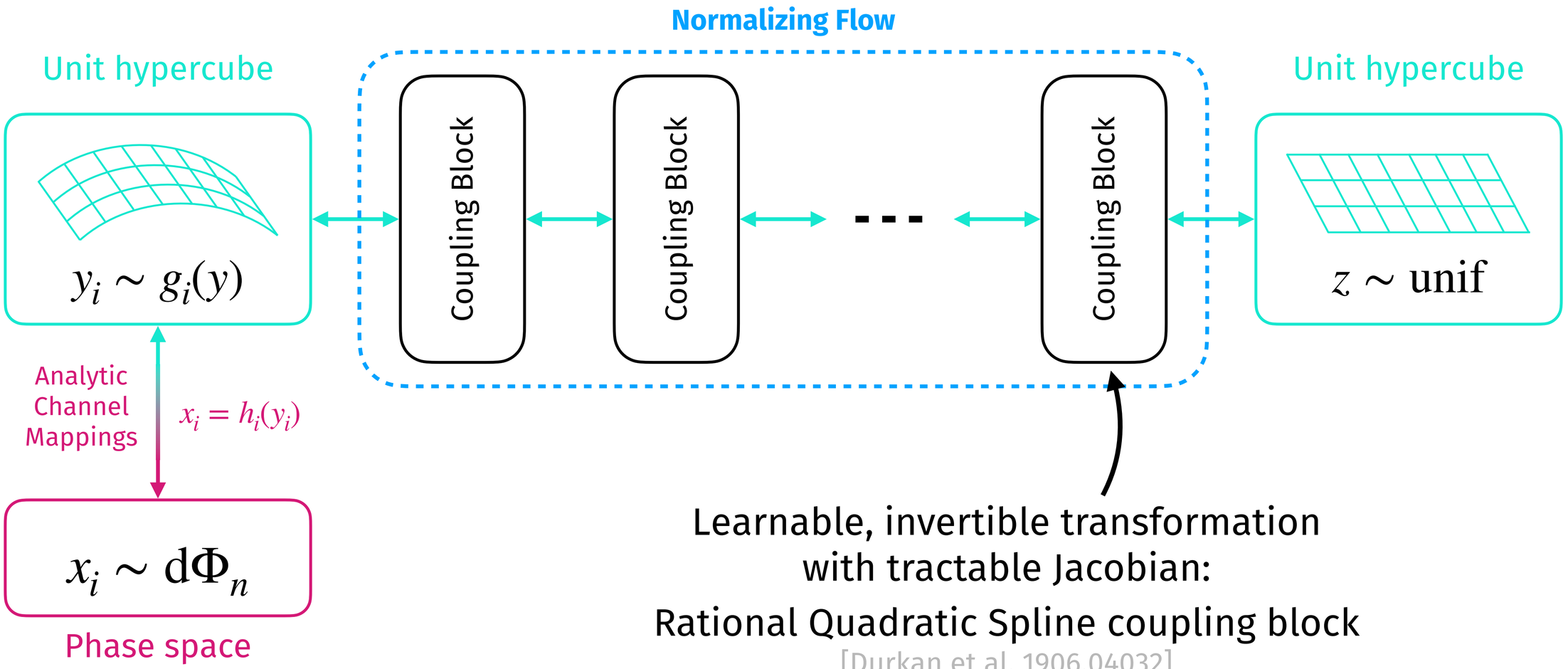
Learned channel mappings

MadGraph: use amplitude structure, ...

Analytic mappings + ~~refine with VEGAS~~

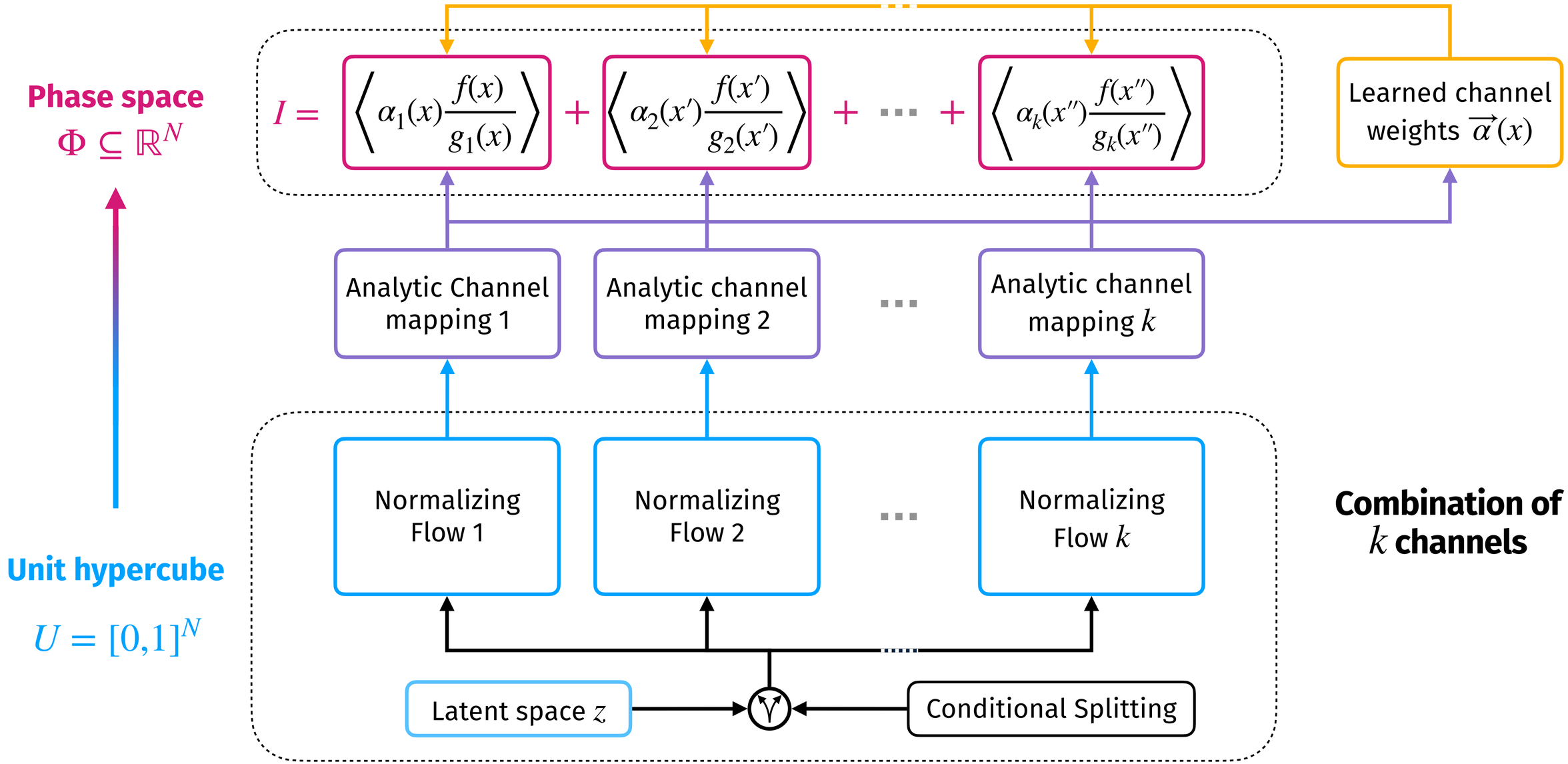
refine with Normalizing Flow

Neural Importance Sampling

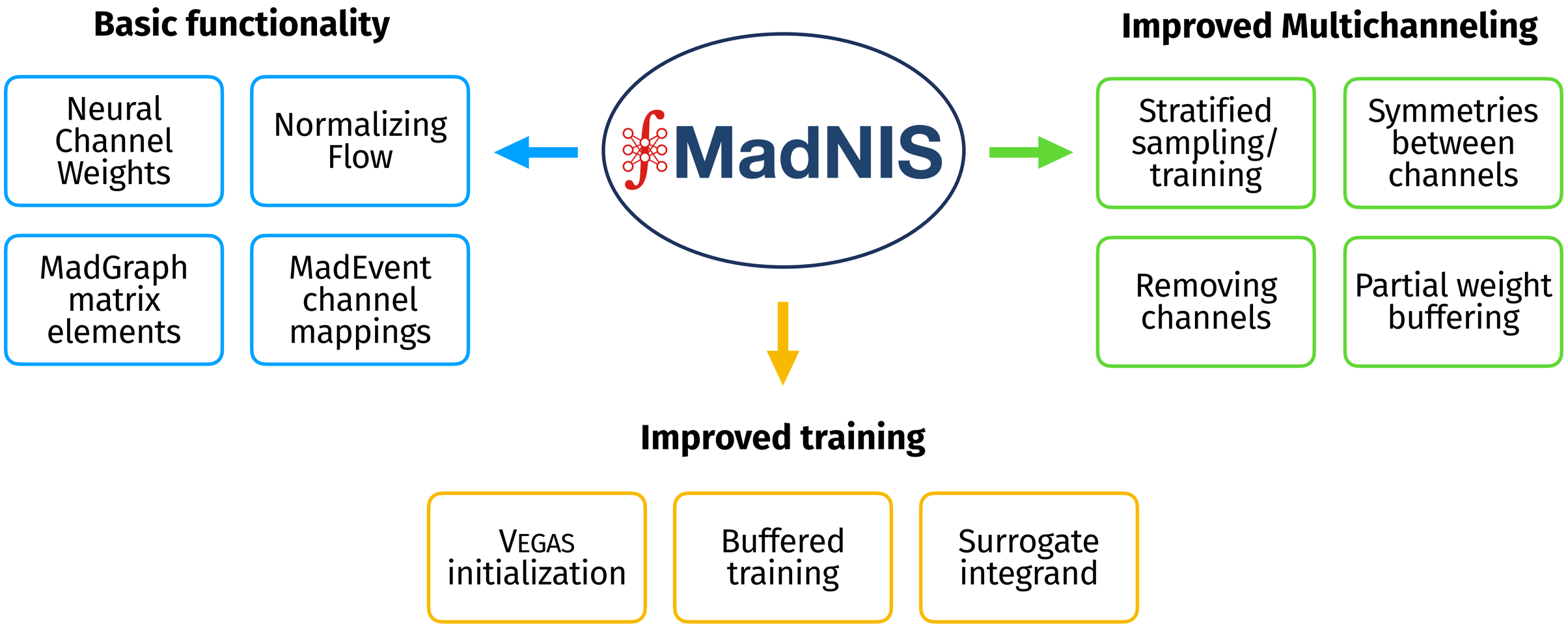


Flows for NIS: [Gao et al, 2001.05486] [Gao et al, 2001.10028] [Bothmann et al, 2001.05478] [Winterhalder et al, 2112.09145]

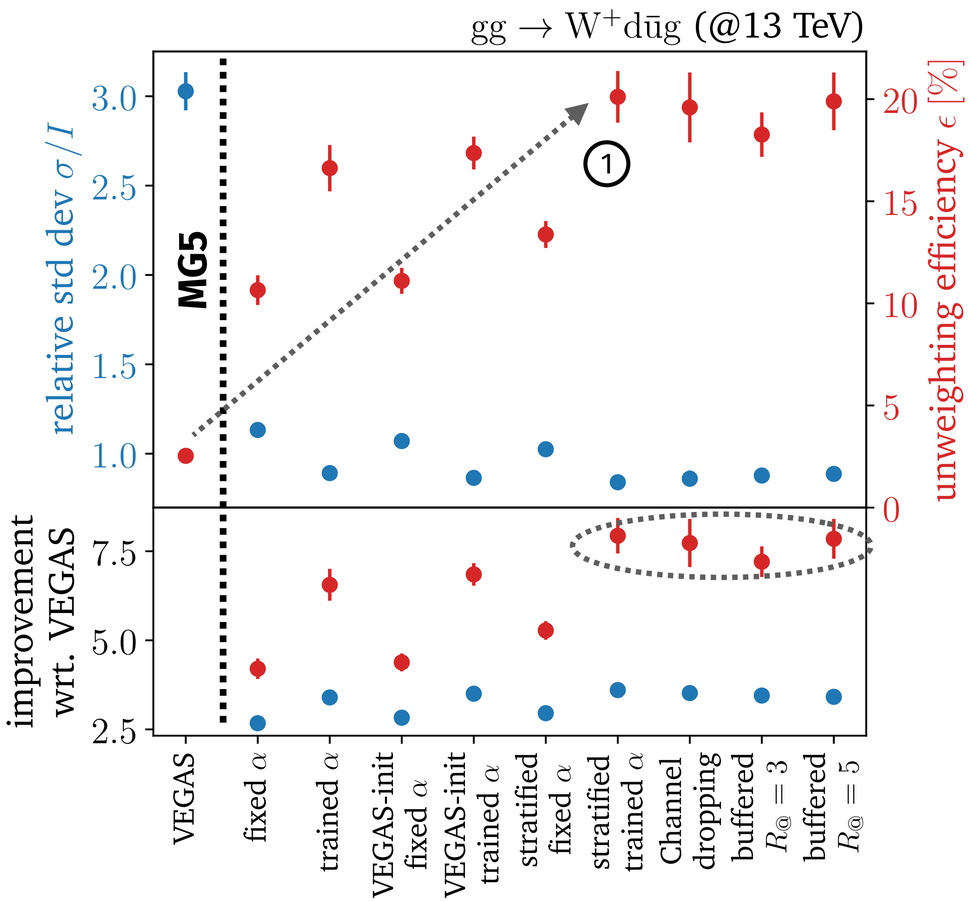
MadNIS: Neural Importance Sampling

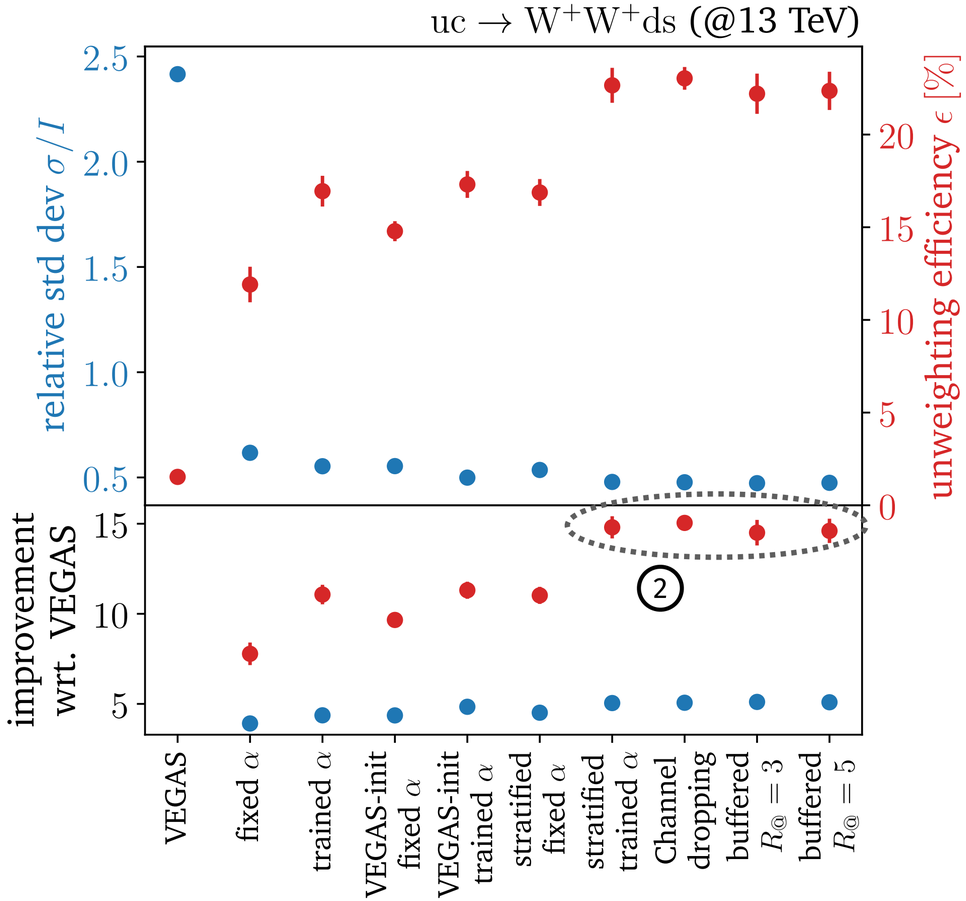


Overview


Basic functionality
Neural Channel Weights
Normalizing Flow
MadGraph matrix elements
MadEvent channel mappings
MadNIS
Improved Multichanneling
Stratified sampling/ training
Symmetries between channels
Removing channels
Partial weight buffering
Improved training
VEGAS initialization
Buffered training
Surrogate integrand

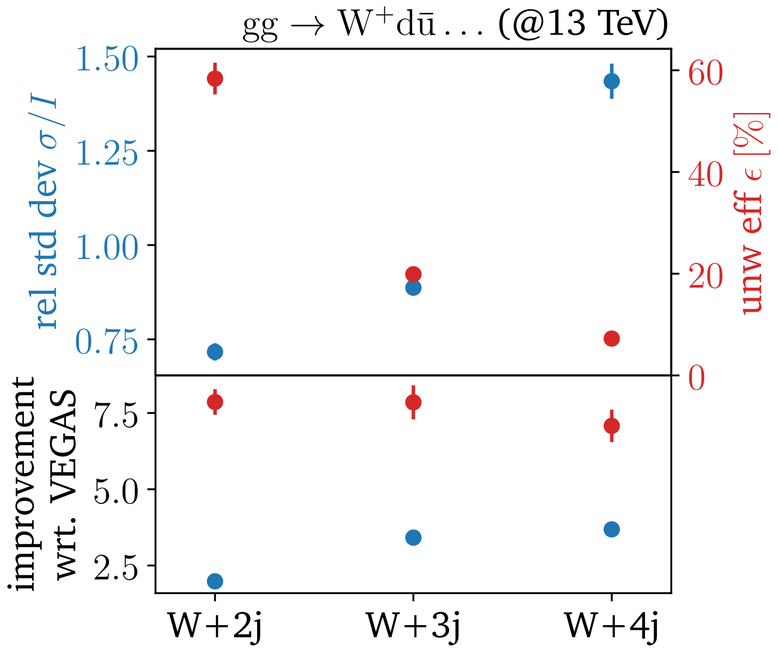
LHC processes

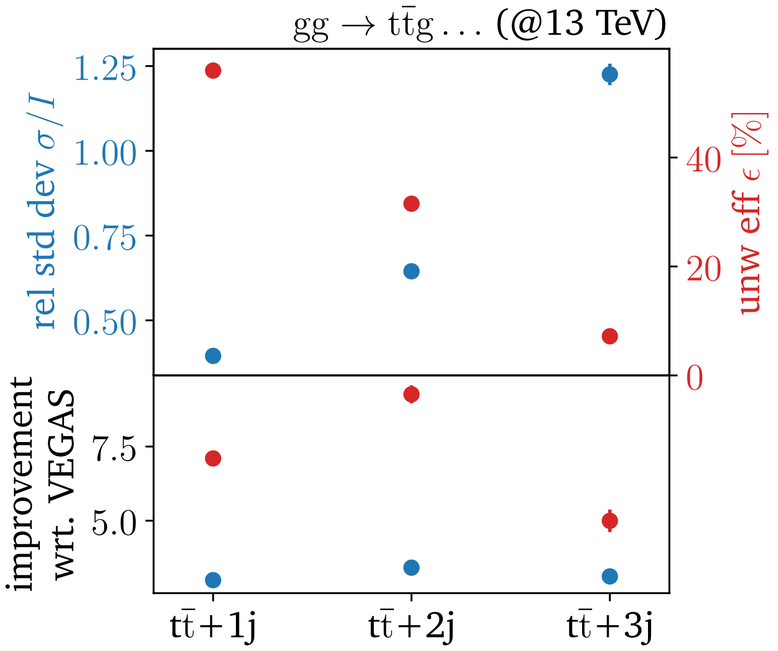




1. Excellent results by combining all improvements!
2. Even larger improvements for process with large interference terms

Scaling with multiplicity





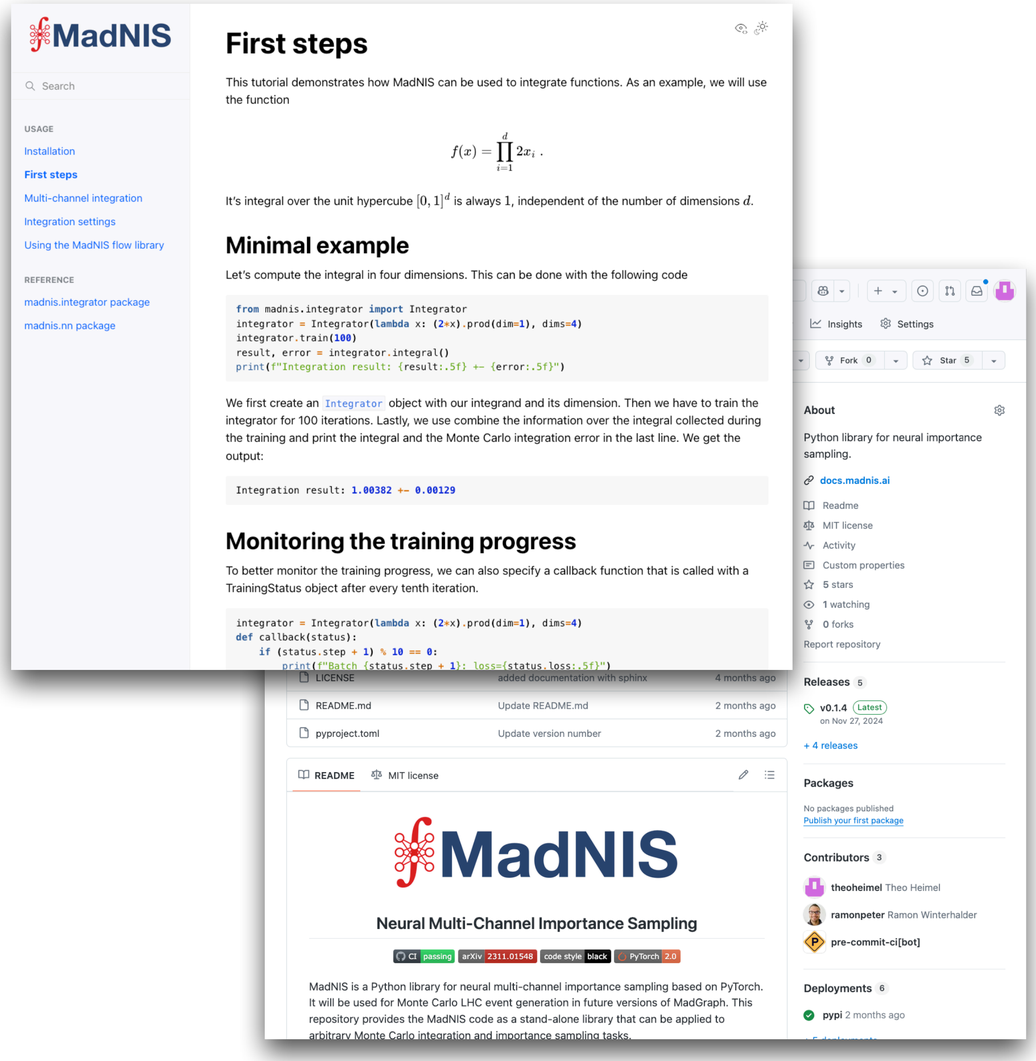
$gg \to W^+d\bar{u}gg$
384 channels, 108 symm.
7x better than VEGAS

$gg \to t\bar{t}ggg$
945 channels, 119 symm.
5x better than VEGAS

Large improvements compared to VEGAS even
for high multiplicities and many channels!

Standalone Python module

- Released MadNIS as a Python package
  → apply to your own integration tasks
- From simple single-channel integrals to complex multi-channel setups

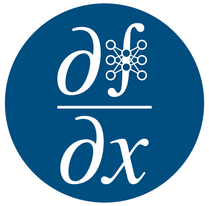


https://madnis.ai/

```
pip install madnis
```

Outline

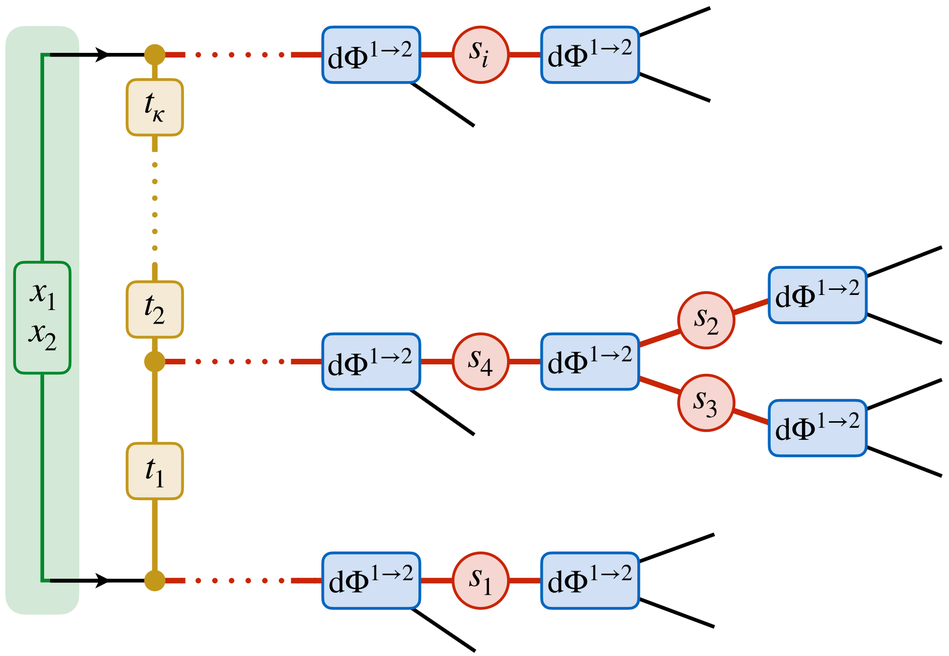
Introduction to MadNIS



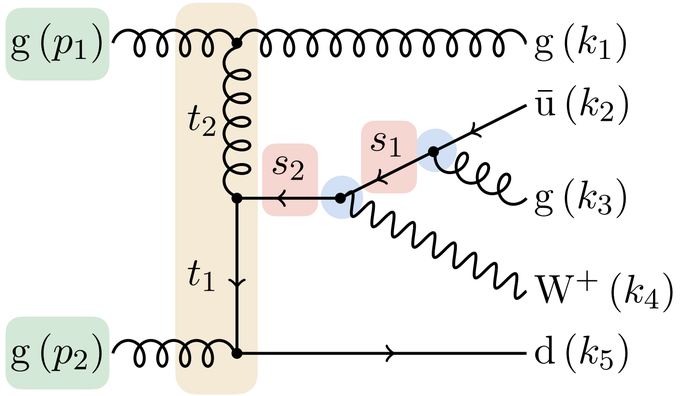
Differentiability and phase space

The road to MadGraph 7

MadNIS-Lite



- Standard construction of PS-mappings from Feynman diagrams

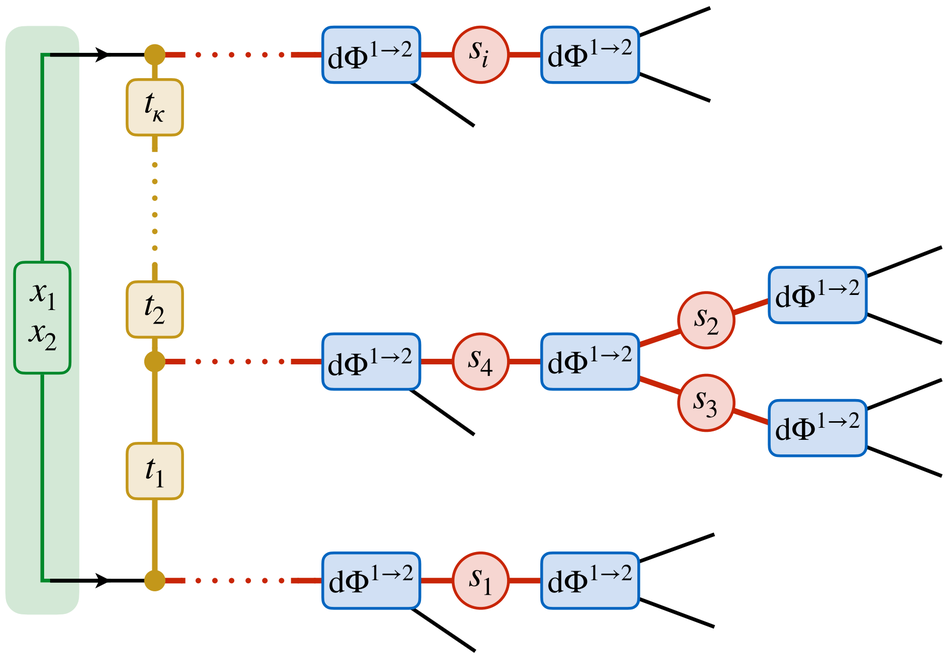


Example:
W+jets

- Phase space library based on PyTorch

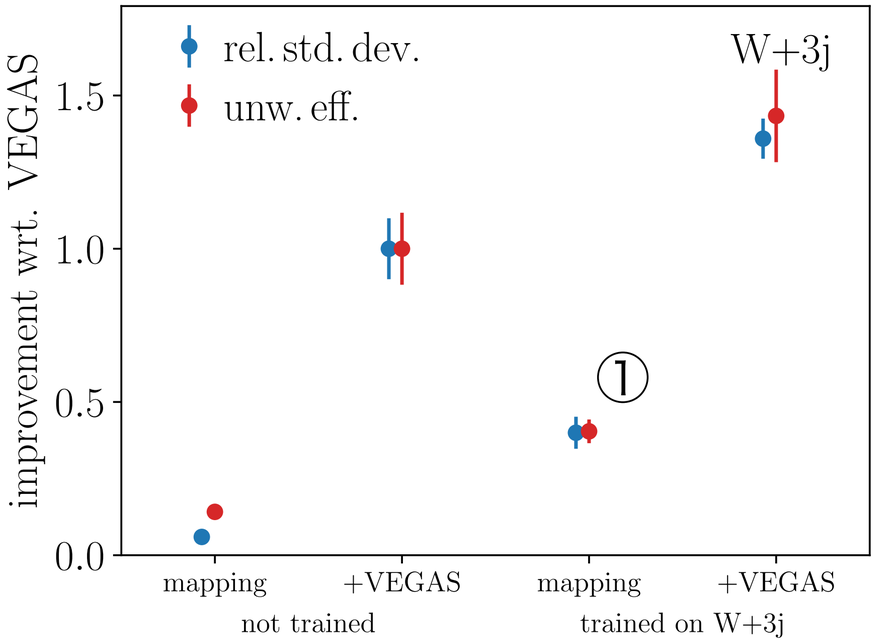
- **Fully differentiable + invertible**
  → can build in trainable components
  → include in MadNIS training

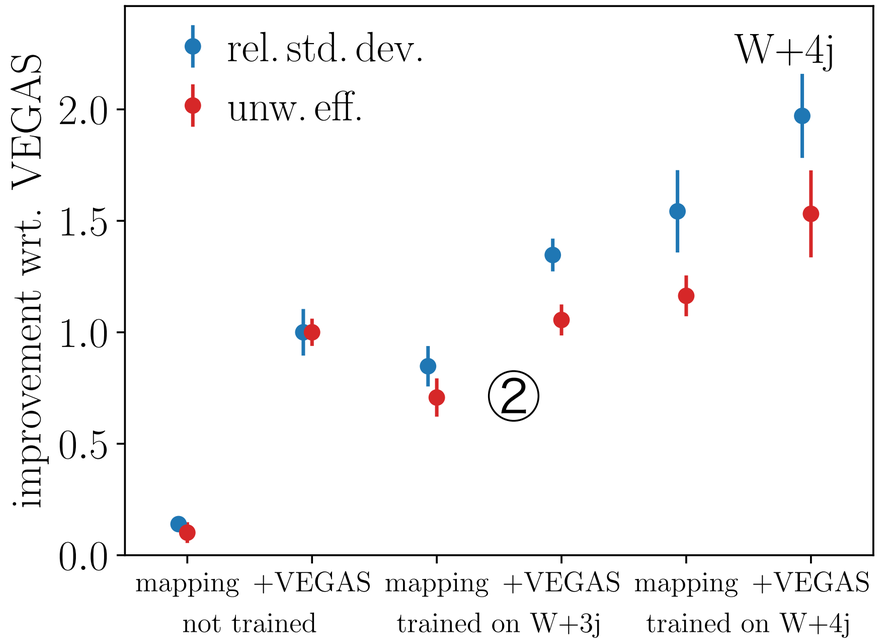
Trainable components



- Add small trainable components based on RQ spline transformations
- Condition on context
  → COM energy, decay energy, …
- **Tiny number of parameter**: shared
  → between all components of same type
  → between all channels

Performance

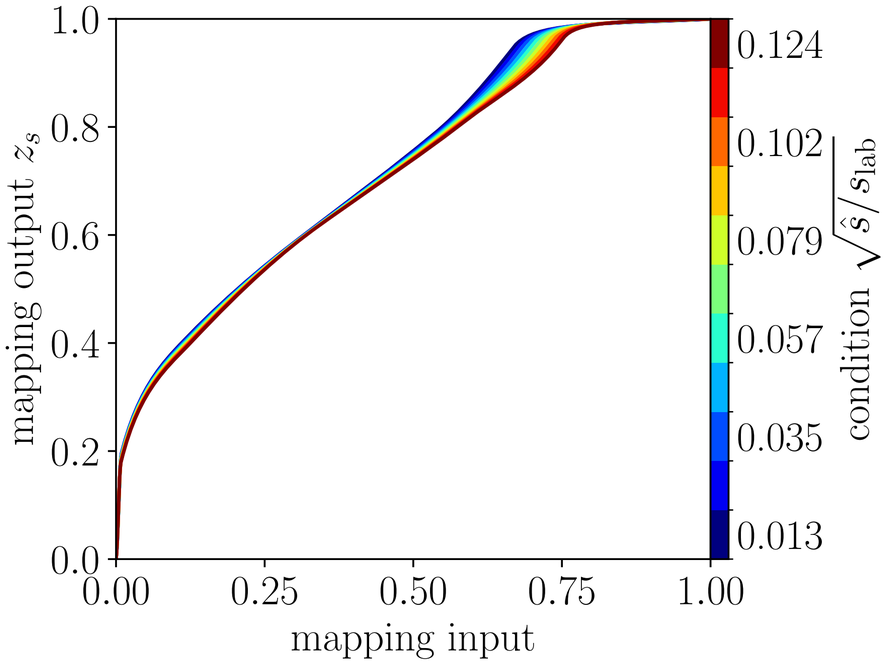




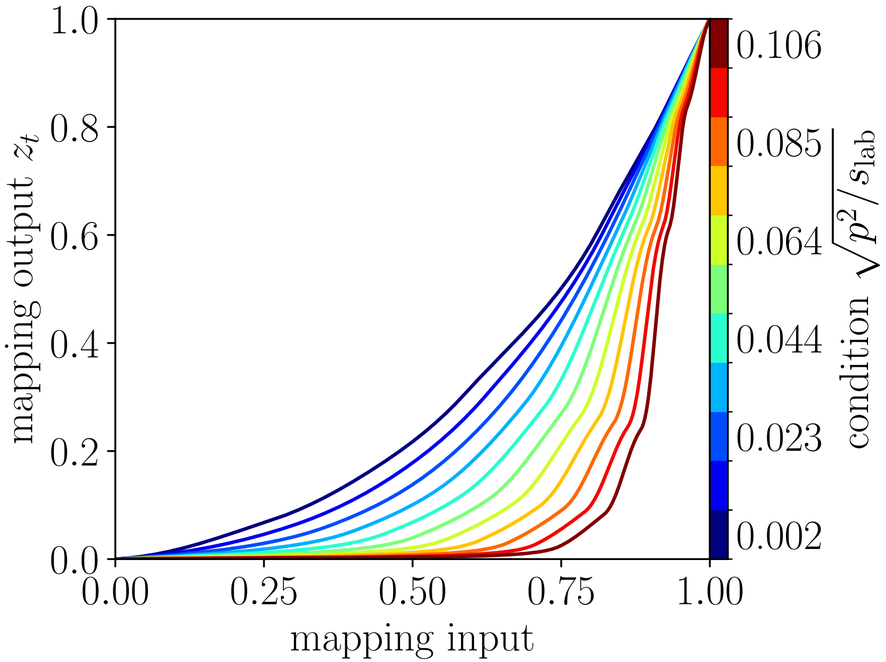
- good performance even though no channel-specific training ①
- **trained for n jets, used for n+1 jets** → performance like VEGAS ②
- further improvements for VEGAS trained on top of MadNIS-Lite

Interpretability

Massless propagator
s-invariant



2→2 scattering
t-invariant



- s-invariant: small energy-dependence, easily learned by VEGAS, still room for improvement in underlying mapping
- t-invariant: large dependence on p^2

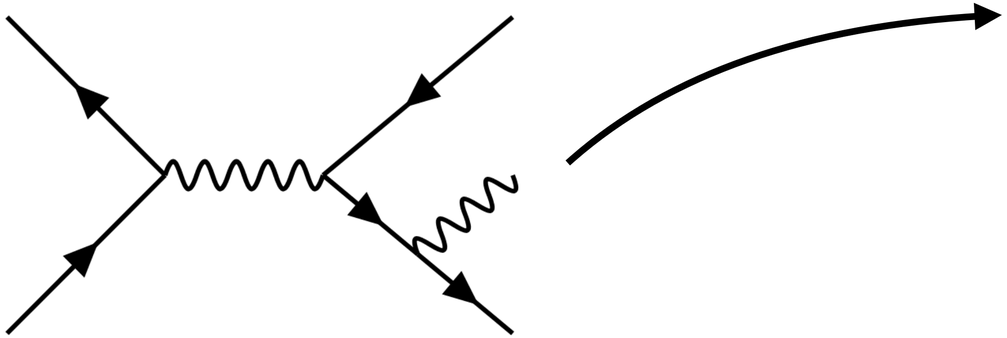
madevent7 phase space library

- New phase space library based on code developed for MadNIS lite
- written in C++/CUDA
  → use SIMD vectorization on CPUs
  → massive parallelization on GPUs
- supported so far:
  → topology-based mappings similar to MG5
  → RAMBO on diet [Plätzer, 1308.2922]
  → CHILI [Bothmann et al., 2302.10449]
- modular structure: easy to incorporate new ideas for phase space

madevent7 phase space library

- Full functionality also available through Python interface
- Accepts inputs from PyTorch, Numpy, …

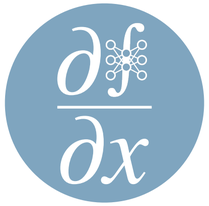
Build a phase space sampler
in 15 lines of Python!



```python
import madevent7 as me
import numpy as np
diagram = me.Diagram(
    incoming_masses=[0., 0.],
    outgoing_masses=[0., 0., 0.],
    propagators=[me.Propagator(91.188, 2.4955),
                 me.Propagator(0., 0.)],
    vertices=[["i0", "i1", "p0"],
              ["p0", "o0", "p1"],
              ["p1", "o1", "o2"]],
)
topology = me.Topology(diagram, me.Topology.all_decays)
mapping = me.PhaseSpaceMapping(topology, 13000.**2)
r = np.random.rand(100000, 7)
(momenta, x1, x2), jac_det = mapping.map_forward([r])
```

Outline

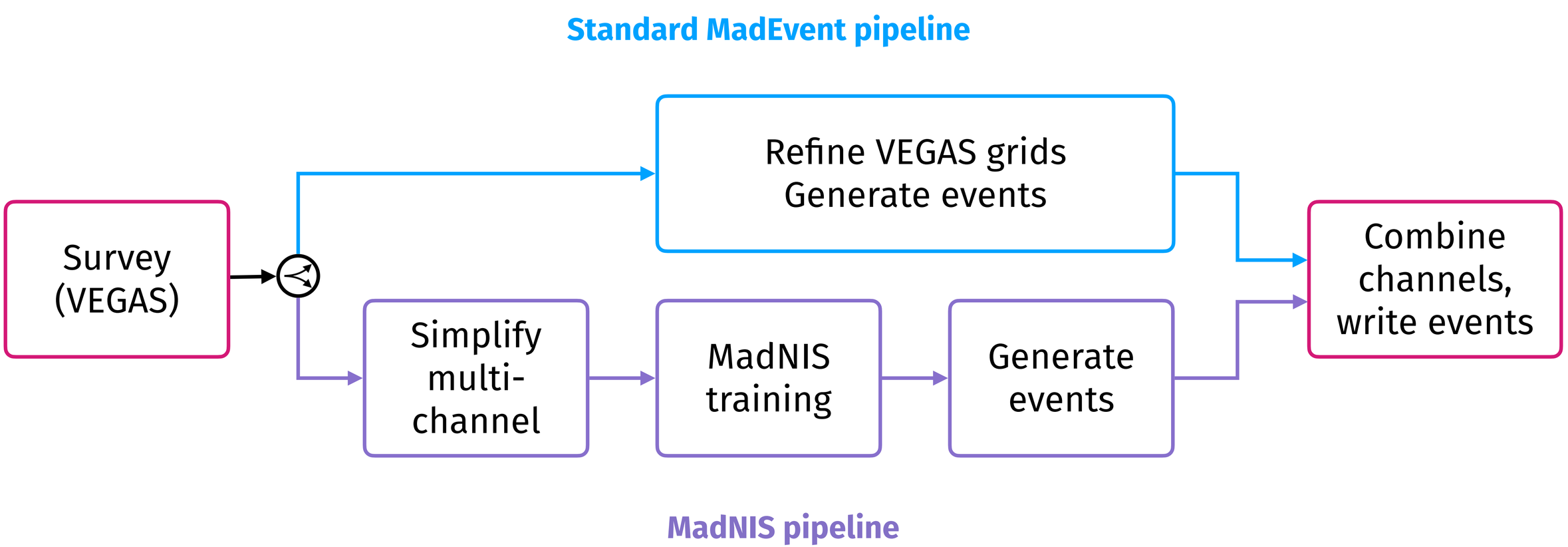
Introduction to MadNIS



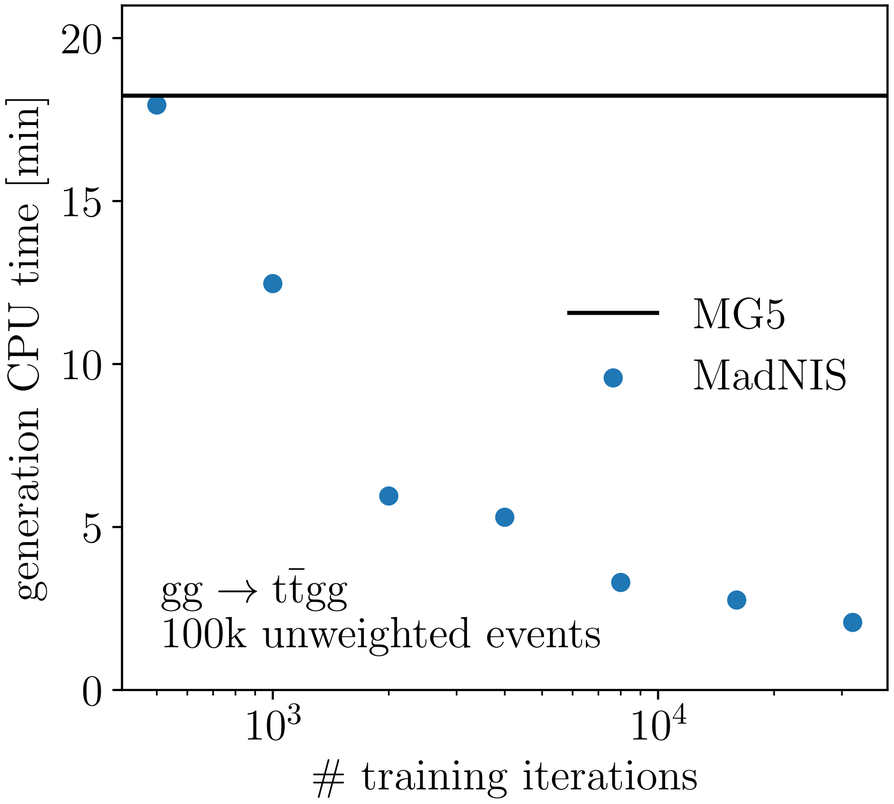
Differentiability and phase space

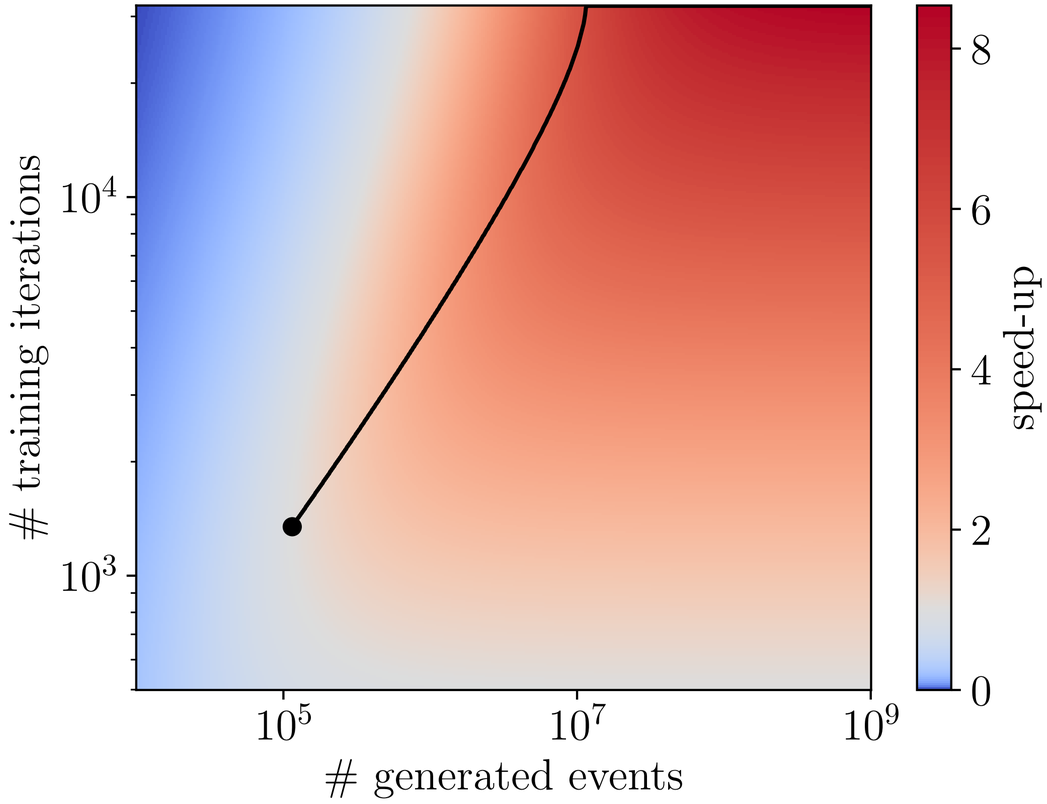
The road to MadGraph 7

Building MadNIS into MadGraph



Example process: gg → tt̄gg

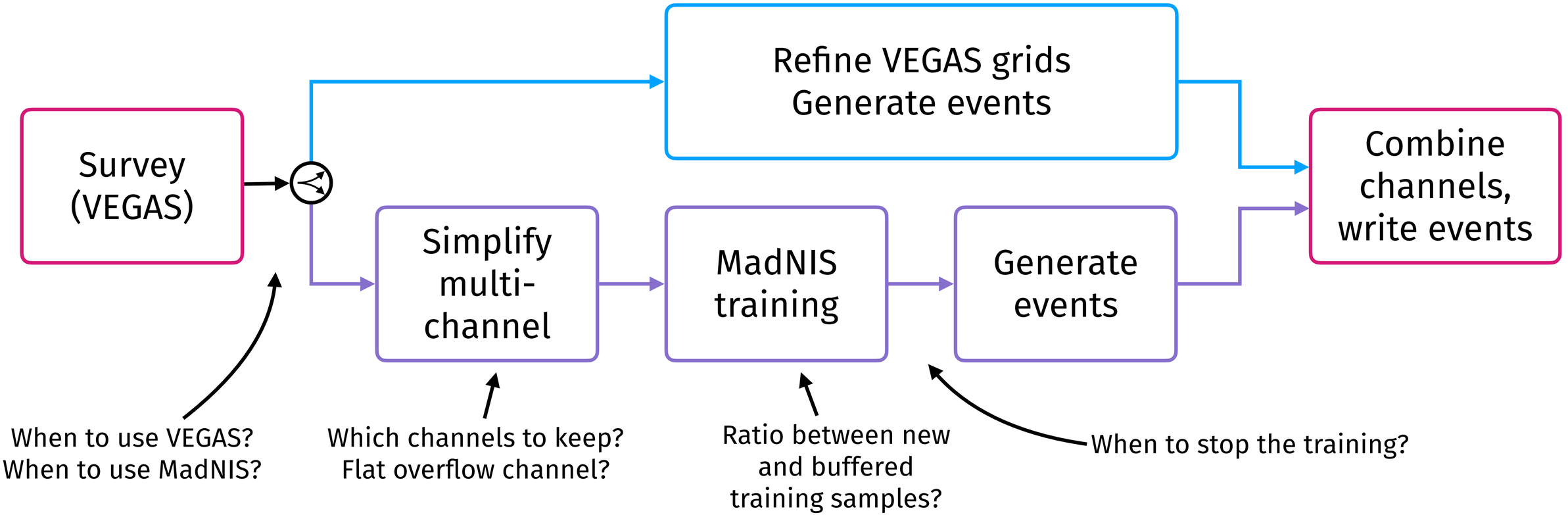




(preliminary)

MadNIS is faster starting at 100k events!

Remaining challenges



Ideally, no tuning should be needed most of the time!

Main challenge: **finding smart defaults and heuristics**

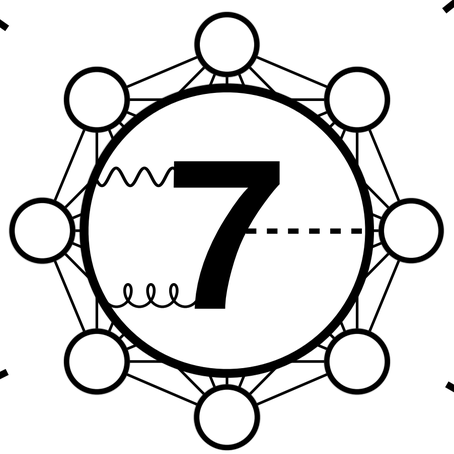
Upcoming MadGraph7

Matrix element on GPU

- huge speed-up from moving to GPU
- improved CPU performance from SIMD vectorization

MadNIS

- neural importance sampling automatically used for most processes
- smart defaults: as easy to use as VEGAS



Faster multi-jet events

- color-aware generator accelerates sampling high-multiplicity processes [2409.12128]

MadEvent7

- new modular phase space generator
- GPU- and ML-enabled
- usable beyond MadGraph

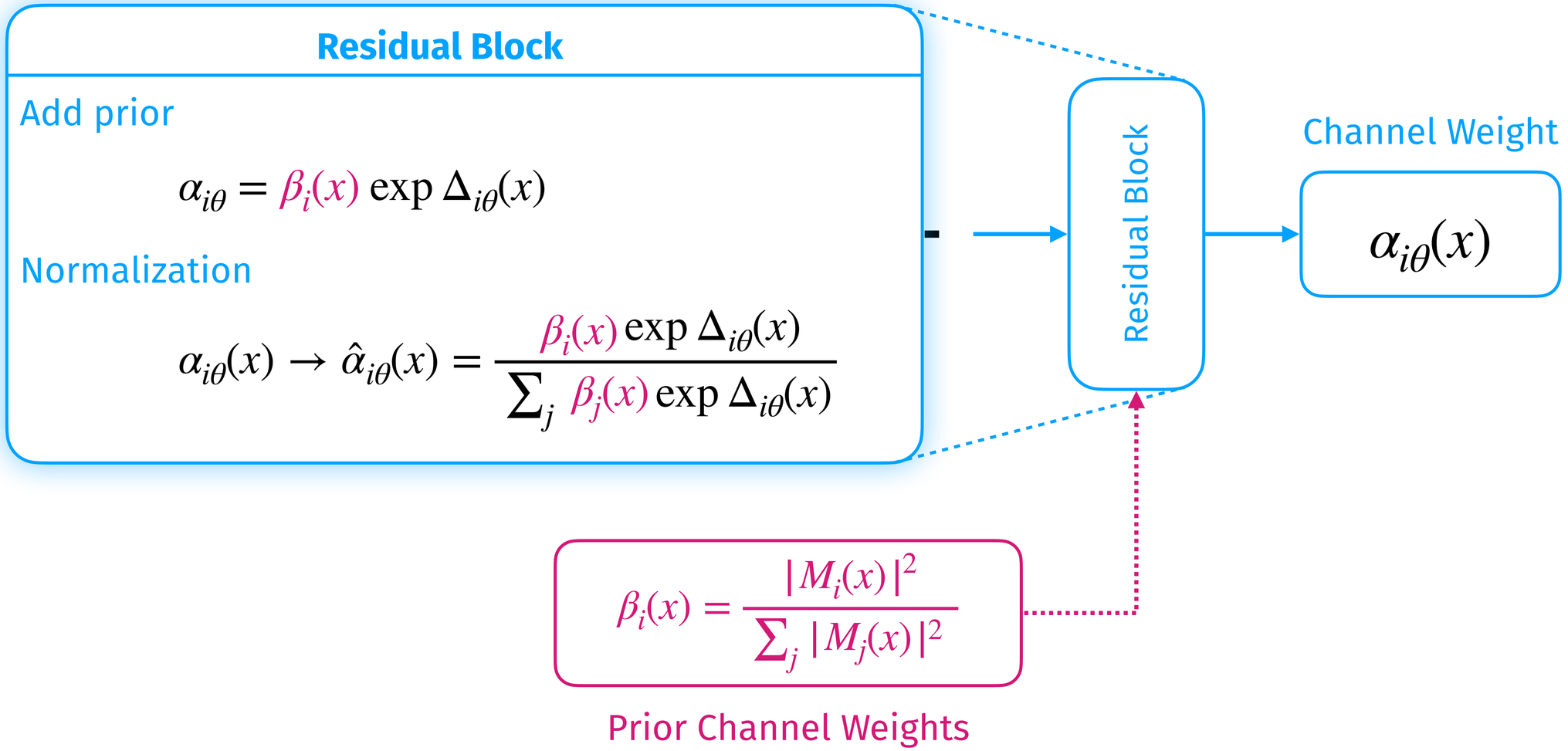
Release planned for the **end of 2025**!

Conclusions

- MadNIS: **improvements in unweighting efficiency** for various realistic LHC examples
- MadNIS-Lite: middle ground between VEGAS and MadNIS
  → **generalizes from n jets to n+1 jets**
  → interpretability to improve phase space mappings
- Speed-up already for low numbers of generated events
- Ongoing work towards MadNIS@NLO
- MadNIS will be an **integral part of upcoming MadGraph 7**!

Appendix

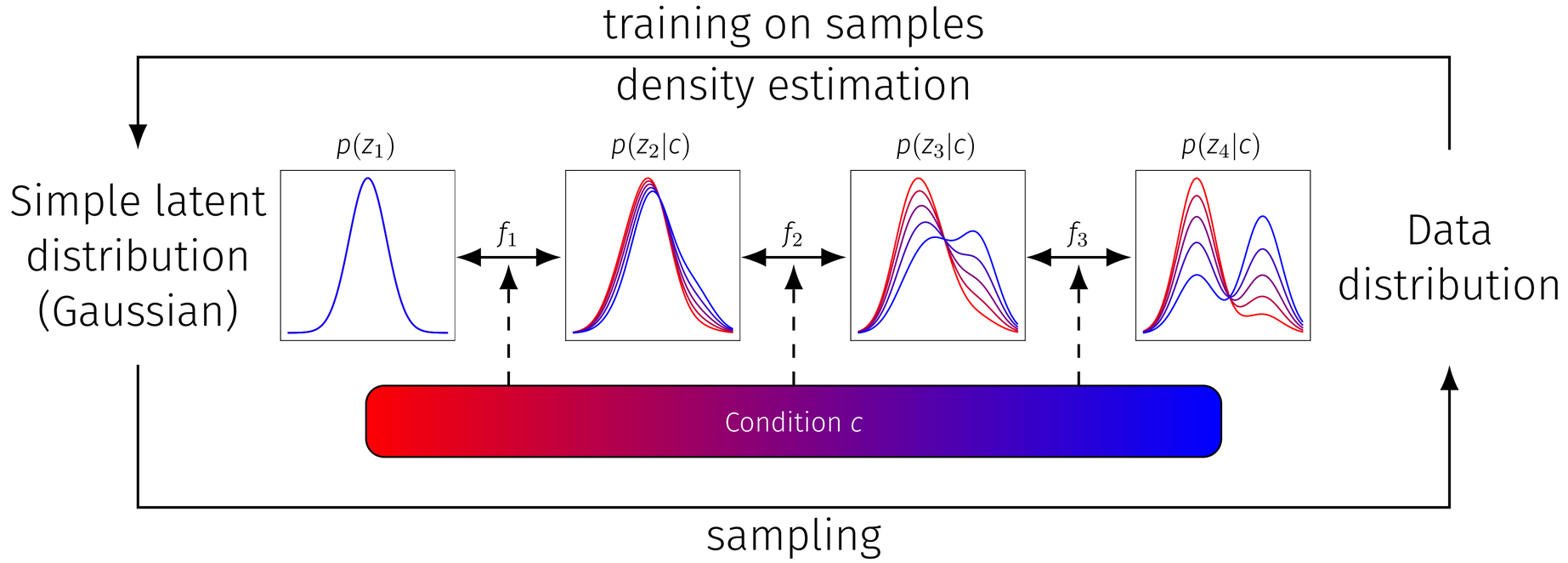
Neural Channel Weights



Normalizing Flows

Chain of **invertible, learnable** transformations with **exact likelihood** from change of variables formula

$$\log p(z_n \mid c) = \log p(z_1) + \log \det \frac{\partial z_1(z_n; c)}{\partial z_n}$$



Flows for NIS: [Gao et al, 2001.05486] [Gao et al, 2001.10028] [Bothmann et al, 2001.05478]

Loss function

Training objective:
Minimize total variance

$$\sigma_{\text{tot}}^2 = N \sum_i \frac{\sigma_i^2}{N_i} \qquad \text{with}$$

$$\sigma_i^2 = \text{Var}\left(\alpha_i(x) \frac{f(x)}{g_i(x)} \right)_{x \sim g_i(x)}$$

Optimal MC weights depend on N_i

↓

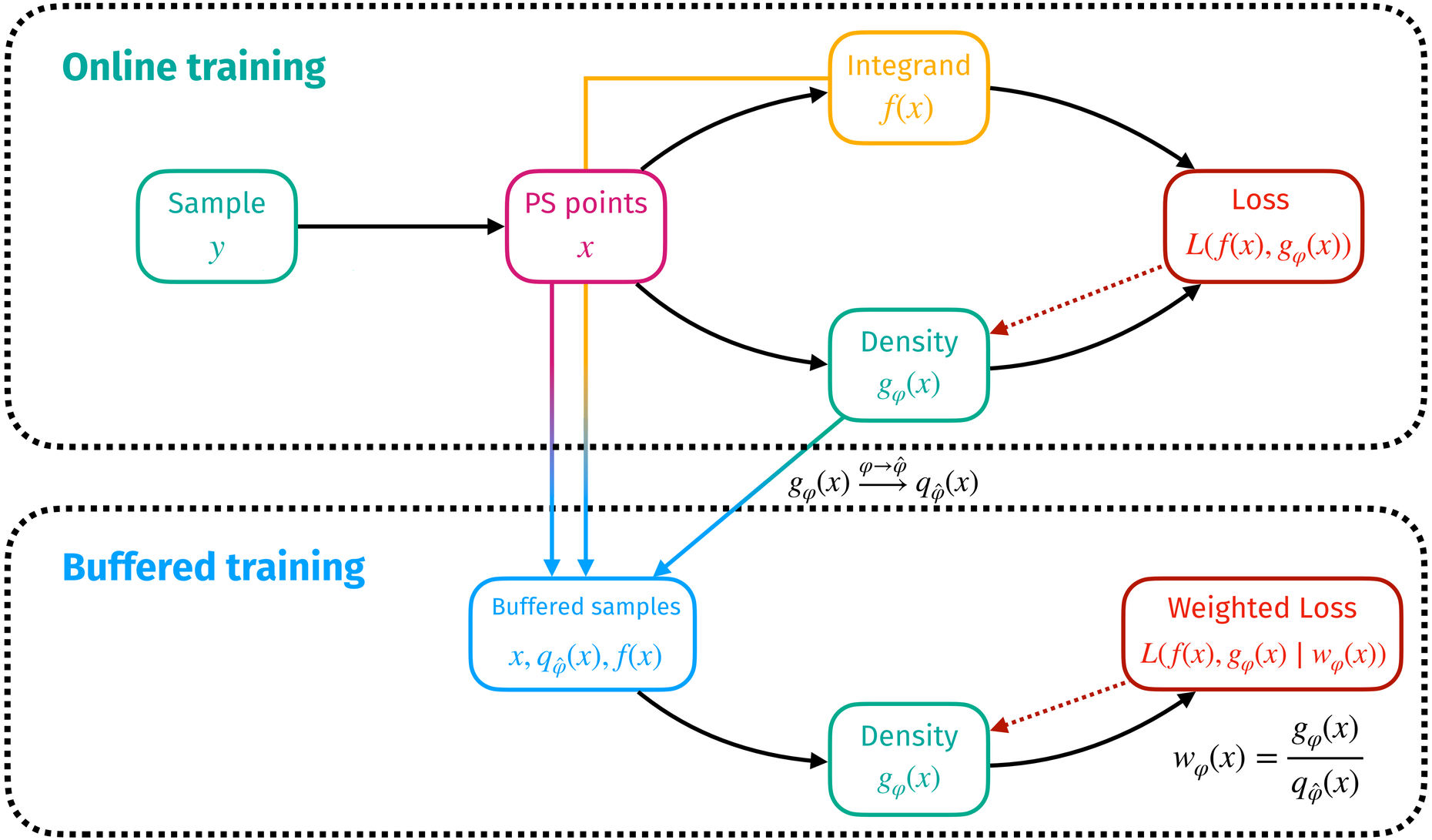
assume choice of N_i during training:
use stratified sampling

$$N_i = N \frac{\sigma_i}{\sum_k \sigma_k}$$

MadNIS loss function

$$\mathscr{L} = \sigma_{\text{tot}}^2 = \sum_{i,k} \sigma_i \, \sigma_k$$

Buffered Training



Buffered Training

Training algorithm

generate new samples, train on them, save samples

↓

train on saved samples n times

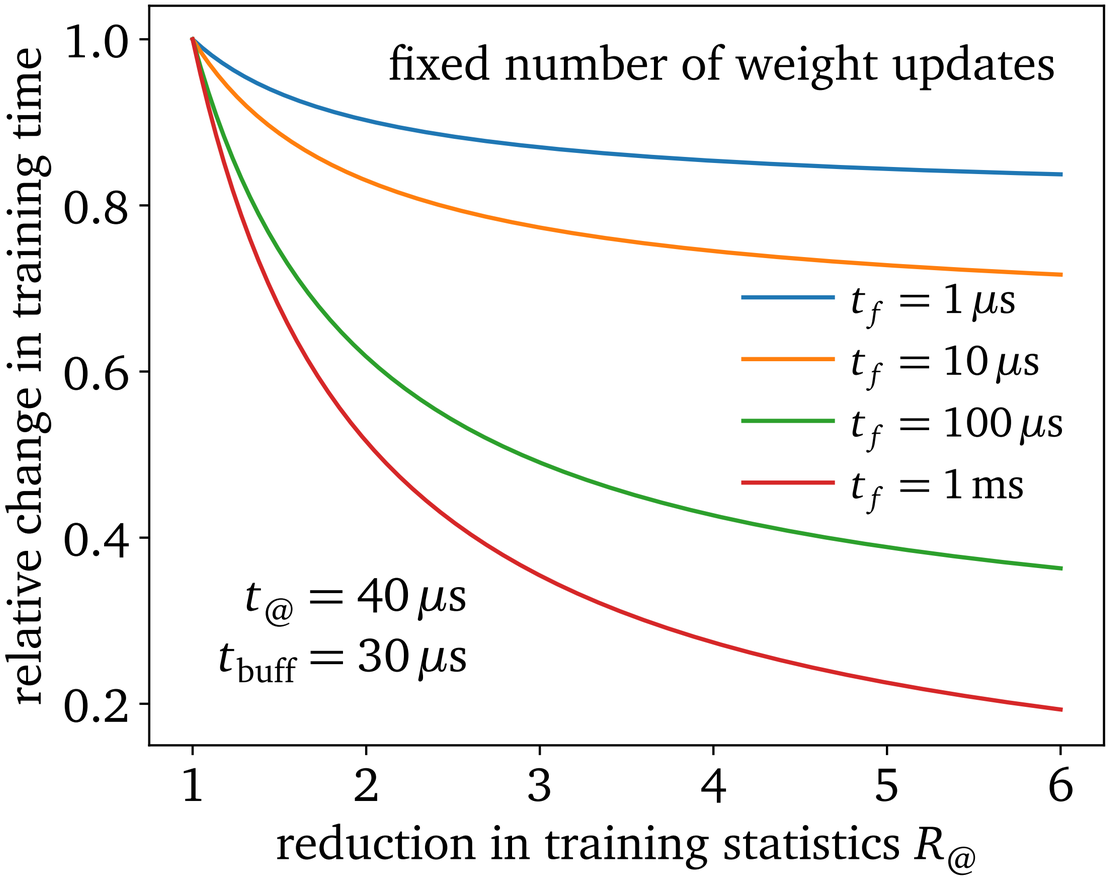
↓

repeat

↓

Reduction in training statistics by

$$R_{@} = n + 1$$

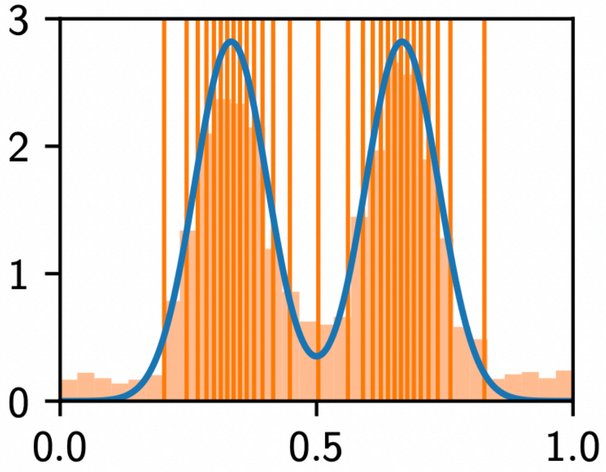


VEGAS algorithm

Factorize probability

$$p(x) = p(x_1) \cdots p(x_n)$$

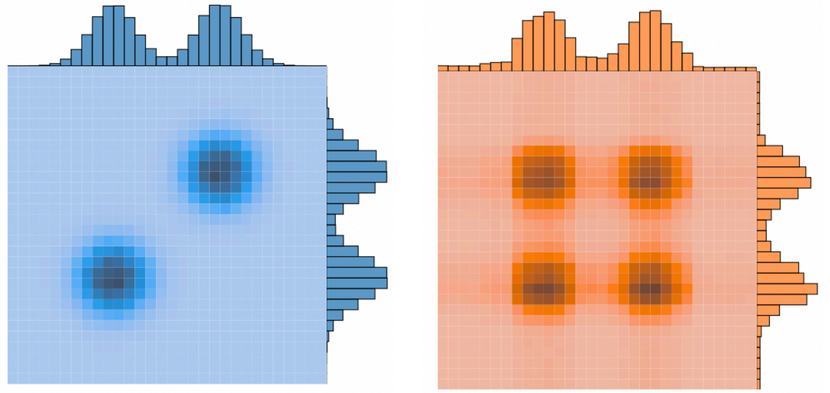
Fit bins with equal probability and varying width



⊕ Computationally cheap

⊖ High-dim and rich peaking functions
→ slow convergence

⊖ Peaks not aligned with grid axes
→ phantom peaks



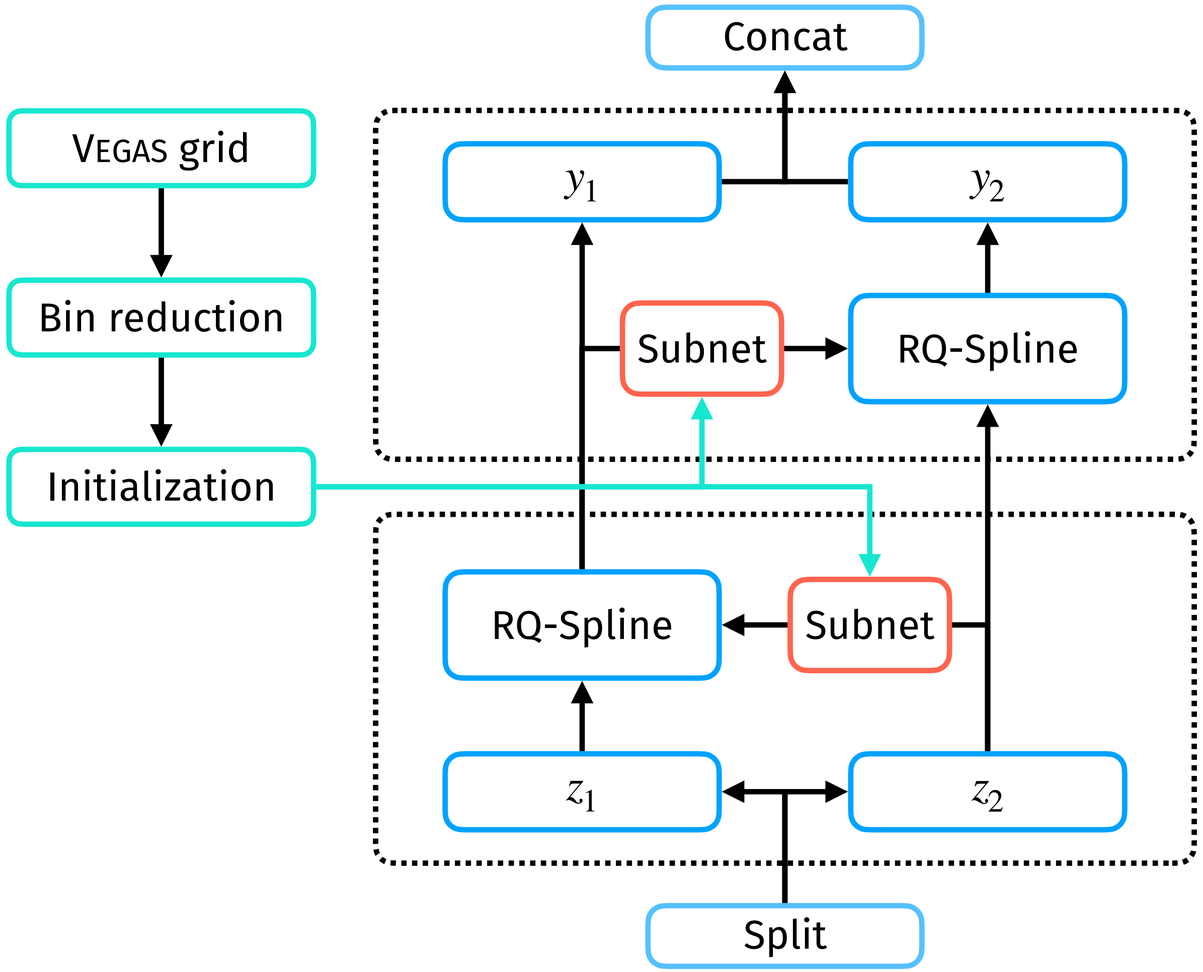
[G. P. Lepage, 1978]

VEGAS Initialization

	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

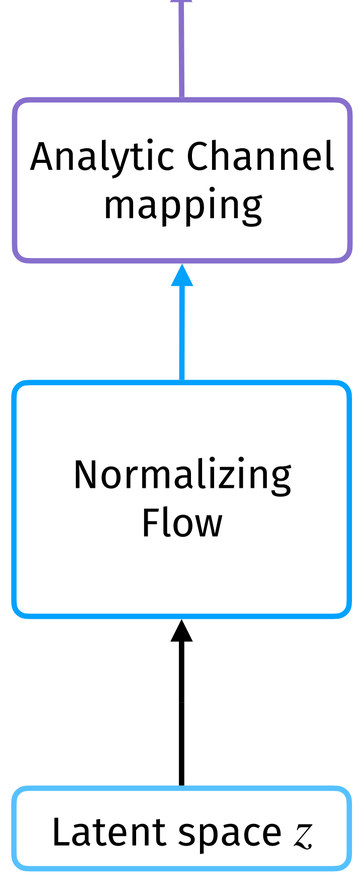
Combine advantages:

Pre-trained VEGAS grid as starting point for flow training



Differentiable MadNIS

Phase space
$\Phi \subseteq \mathbb{R}^N$



Unit hypercube
$U = [0,1]^N$

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\Phi} = \frac{1}{\text{flux}} f(x_a) f(x_b) \langle\, |M_{\lambda,c,\ldots}(p_a, p_b \mid p_1, \ldots, p_n)|^2 \,\rangle$$

Differentiable phase space

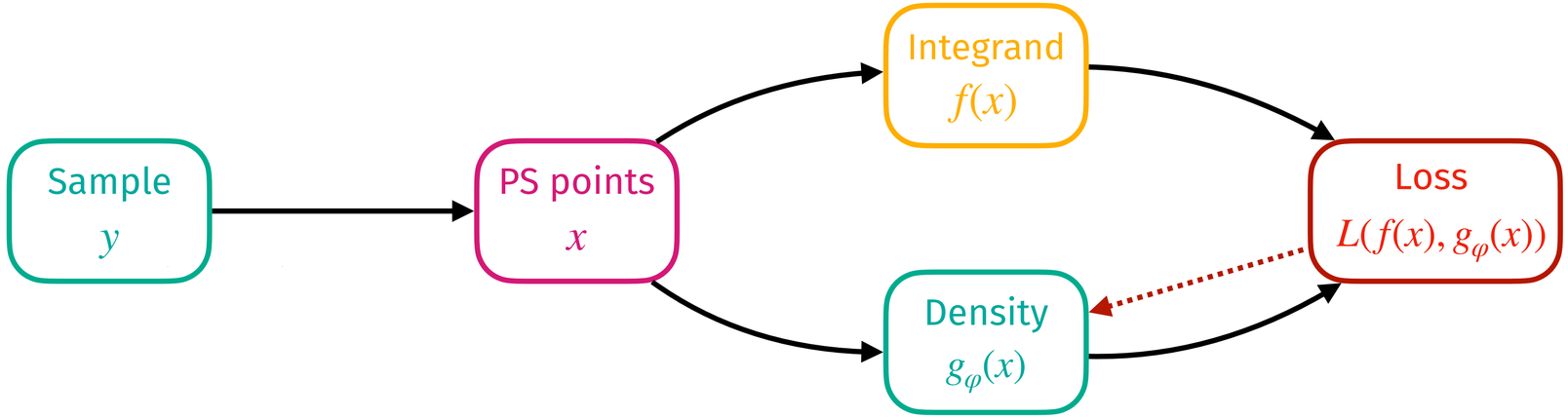
Differentiable parton densities

Differentiable matrix elements

Fully differentiable MadGraph using PyTorch

Based on previous implementations in Jax and TF:
PDFFlow [2009.06635], MadFlow [2106.10279],
MadJax [2203.00057]

MadNIS Training



Forward loss

Regular MadNIS training

$$L_F^{\text{fw}} = \left\langle \frac{g_\theta(x)}{q(x)} F\left(\frac{f(x)}{g_\theta(x)}\right) \right\rangle_{x\sim q(x)}$$

Inverse loss

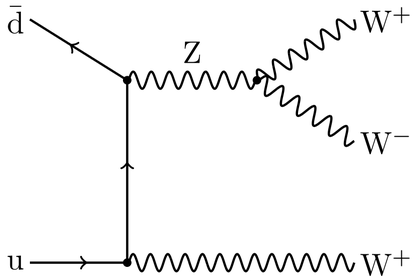
Only possible with differentiable integrand

$$L_F^{\text{inv}} = \left\langle F\left(\frac{f(\overline{G}_\theta(z))}{\overline{g}_\theta(z)}\right) \right\rangle_{z\sim p_0(z)}$$

Arbitrary f-divergence: KL, RKL, variance

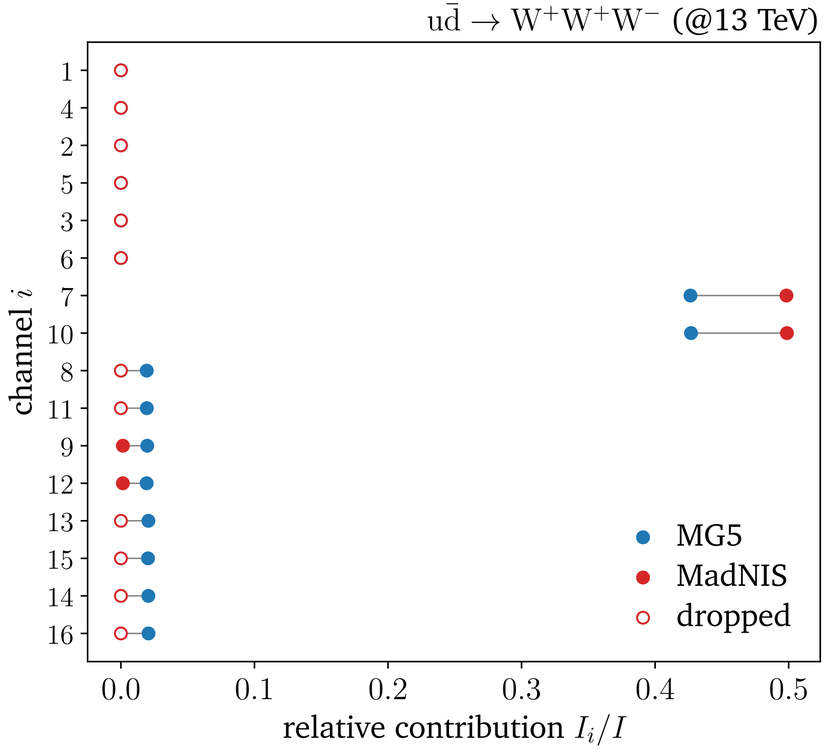
Example process

- Example process: $u\bar{d} \to W^+W^+W^-$

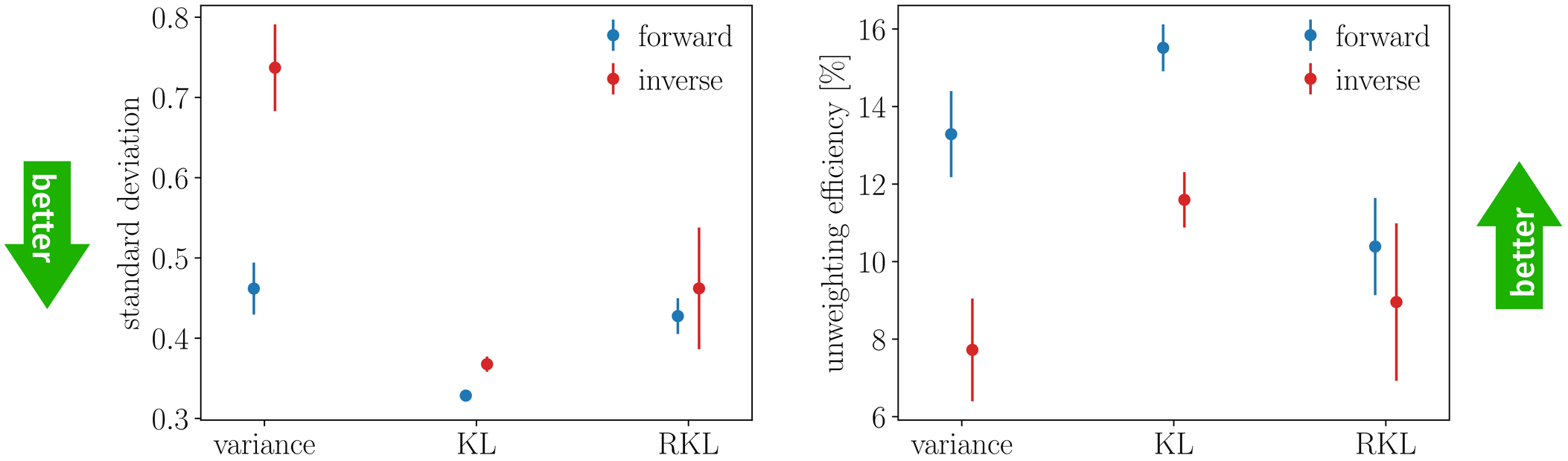


- Last MadNIS paper:
  single channel is sufficient

- Flat, differentiable phase space
  using RAMBO on diet

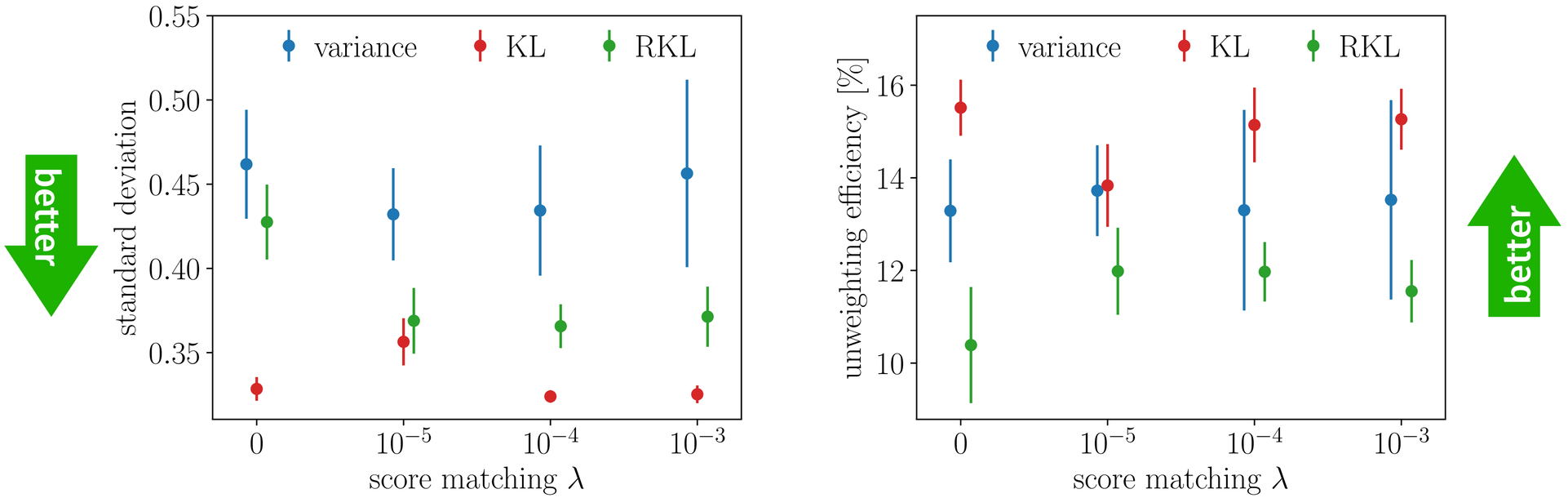


Forward vs inverse loss



- forward losses perform better than inverse losses
- Side result: KL performs better than variance,
  however not compatible with trainable channel mappings

Derivative matching loss

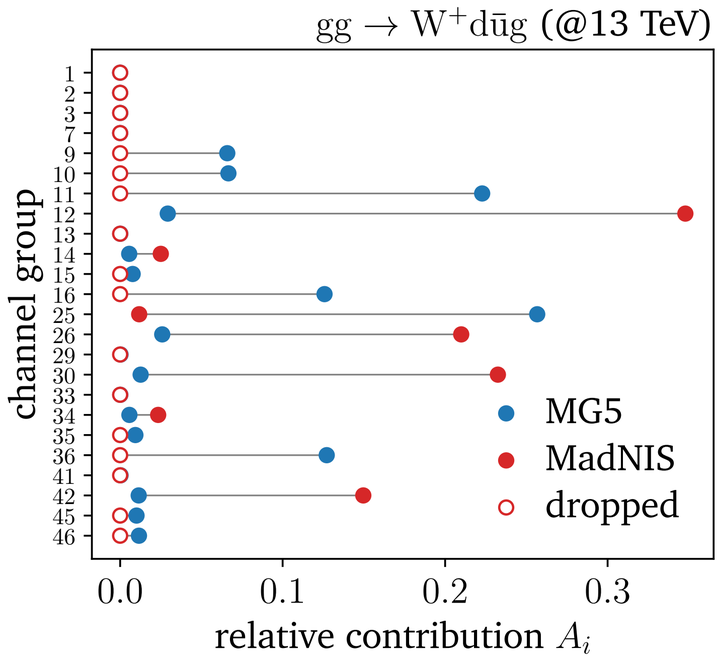


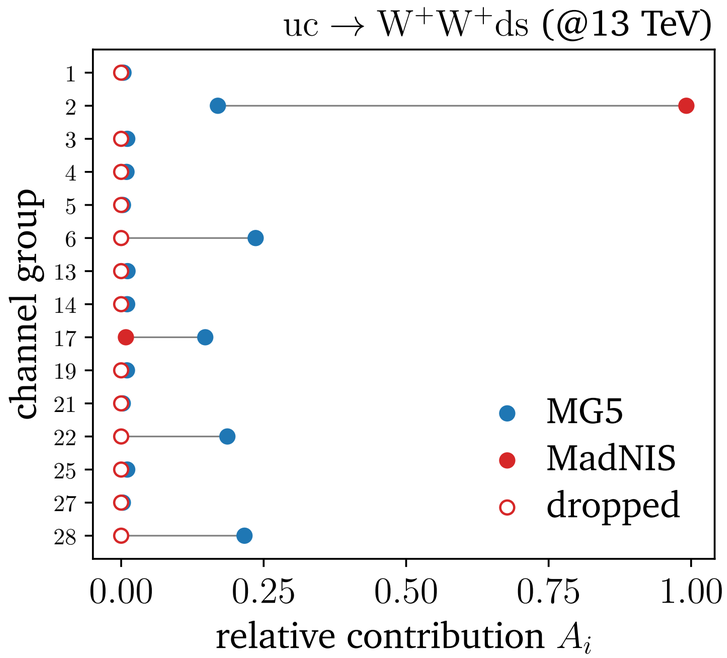
- Alternative use for gradients: derivative matching loss

$$L^{\text{fw}} \to L^{\text{fw}} + \lambda \left\langle \left| \partial_x \log f(x) - \partial_x \log g_\theta(x) \right|^2 \right\rangle_{x \sim q(x)}$$

- Sometimes small improvements over normal forward loss
- Additional **cost of gradient evaluation not amortized**

Learned channel weights



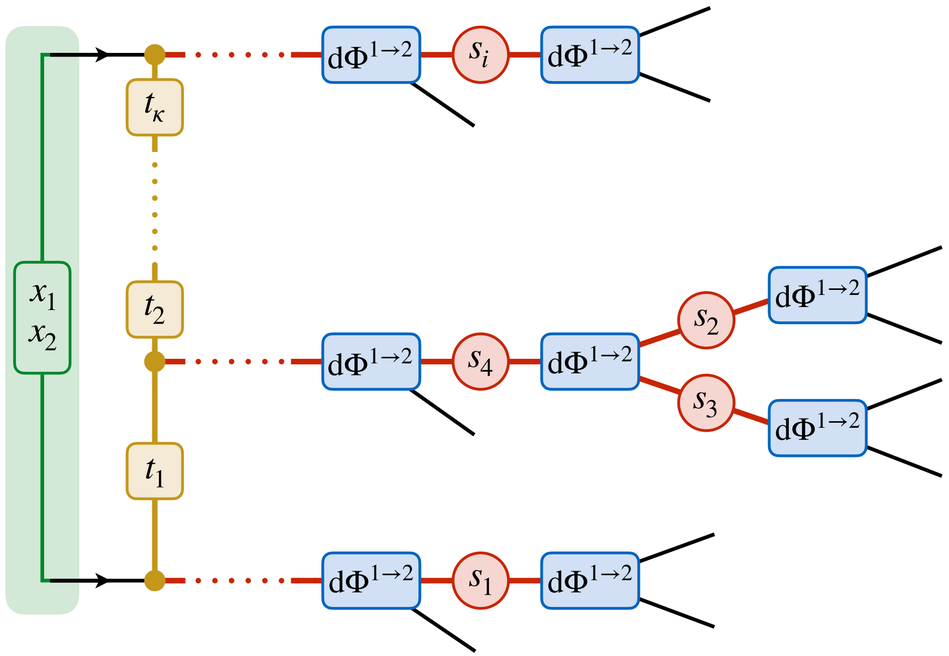


MadNIS often sends weight of many channels to 0

↓

dropping channels makes training and
event generation more stable and efficient

Trainable components



Mapping	Parameters	Conditions
Time-like invariants, Eqs.(39),(40) (separate for massless and massive propagators)	190	partonic CM energy $\sqrt{\hat{s}/s_{\text{lab}}}$ minimal decay CM energy $\sqrt{s_{\text{min}}/s_{\text{lab}}}$ maximal decay CM energy $\sqrt{s_{\text{max}}/s_{\text{lab}}}$
$2 \to 2$ scattering, Eq.(43)	798	correlations between z_t, z_ϕ partonic CM energy $\sqrt{\hat{s}/s_{\text{lab}}}$ scattering CM energy $\sqrt{p^2/s_{\text{lab}}}$ virtualities $\sqrt{k^2_{1,2}/s_{\text{lab}}}$
Time-like invariants for pseudo-particles, Eq.(45)	190	partonic CM energy $\sqrt{\hat{s}/s_{\text{lab}}}$ minimal energy $\sqrt{s_{\text{min}}/s_{\text{lab}}}$ maximal energy $\sqrt{s_{\text{max}}/s_{\text{lab}}}$
$1 \to 2$ decay, Eq.(46)	380	correlations between z_θ, z_ϕ partonic CM energy $\sqrt{\hat{s}/s_{\text{lab}}}$ decay CM energy $\sqrt{p^2/s_{\text{lab}}}$
PDF convolutions, Eq.(48)	114	correlations between z_τ, z_{x_1}

Total: 1672 parameters