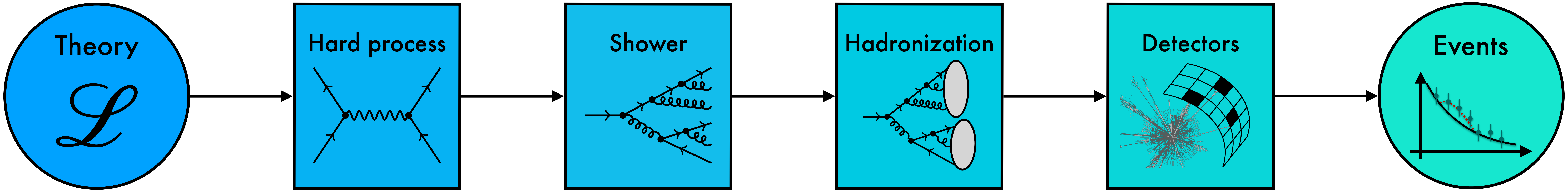
MadNIS — MadGraph Neural Importance Sampling

Theo Heimel
November 2024

[2212.06172] TH, Winterhalder, Butter, Isaacson, Krause, Maltoni, Mattelaer, Plehn
[2311.01548] TH, Huetsch, Maltoni, Mattelaer, Plehn, Winterhalder
[2408.01486] TH, Mattelaer, Plehn, Winterhalder
[2411.00942] TH, Plehn, Schmal

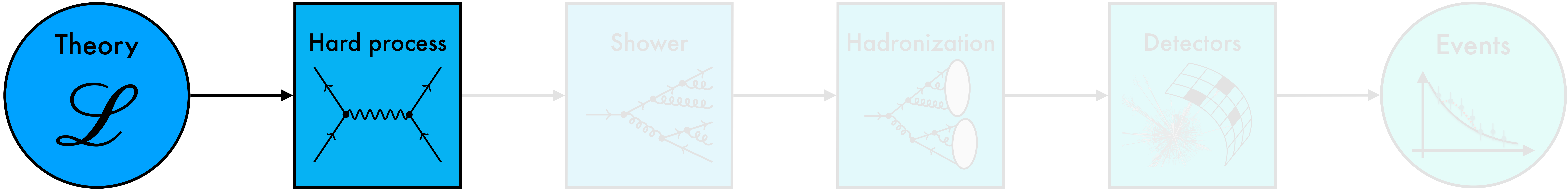
Introduction

How can we prevent MC event generation from becoming a bottleneck in future LHC runs?



Introduction

How can we prevent MC event generation from becoming a bottleneck in future LHC runs?



Differential cross section known from QFT:

$$d\sigma \sim \mathrm{pdf}(x) \cdot |\mathscr{M}(x)|^2 \cdot d\Phi$$

Total cross section:

$$\sigma = \int_\Phi d\sigma$$

Monte Carlo integration and sampling from differential cross section

↓

accelerate with deep generative models

Exact sampling ensured by known likelihood

↓

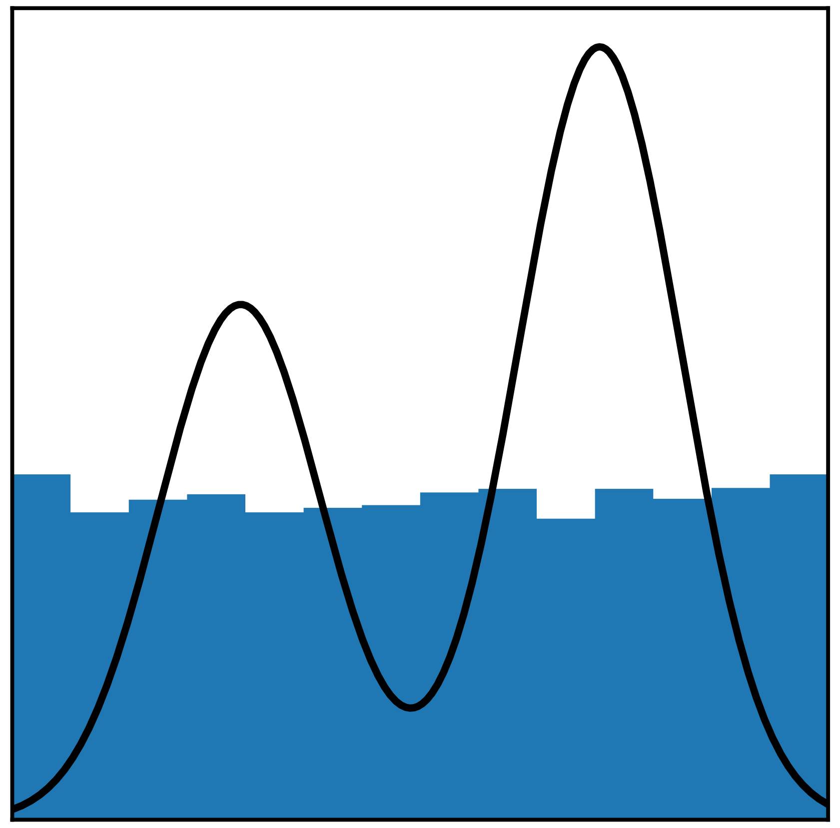
better model
=
faster sampling

Monte Carlo integration + sampling

$$I = \int \mathrm{d}x\, f(x)$$

Monte Carlo integration + sampling

$$I = \int \mathrm{d}x\, f(x)$$

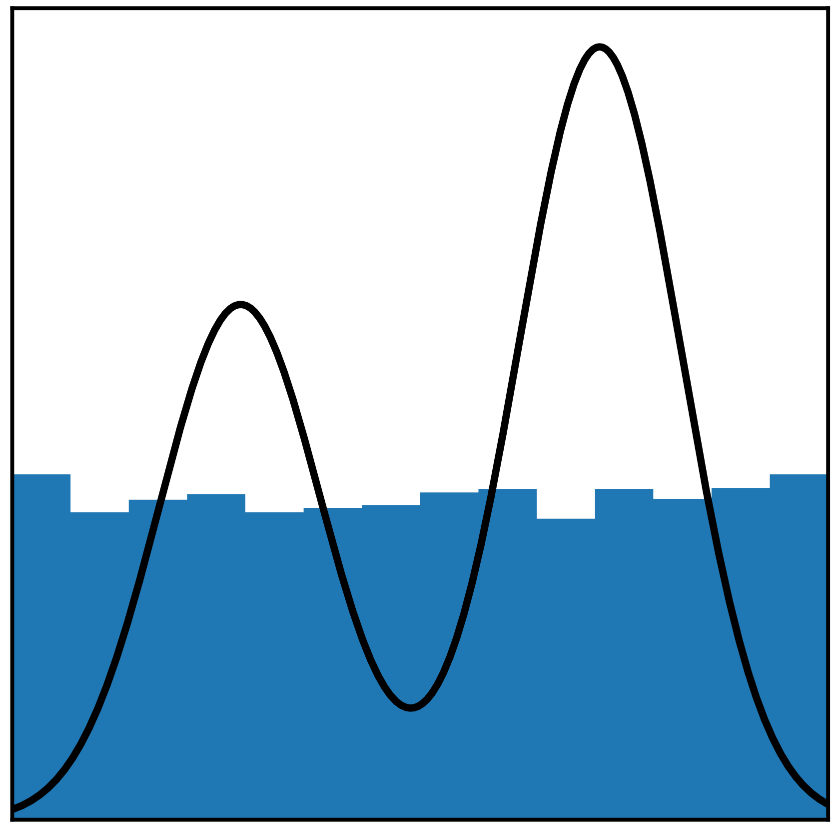


Flat sampling
inefficient

$$I = \langle f(x) \rangle_{x \sim p(x)}$$

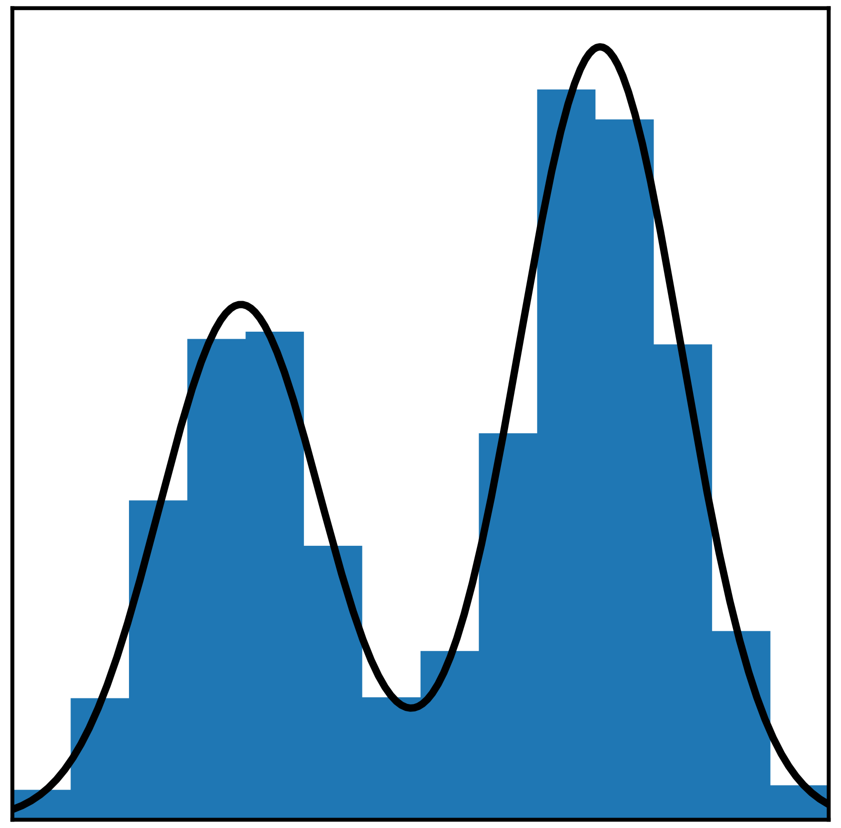
Monte Carlo integration + sampling

$$I = \int \mathrm{d}x\, f(x)$$



Flat sampling
inefficient

$$I = \langle f(x) \rangle_{x \sim p(x)}$$

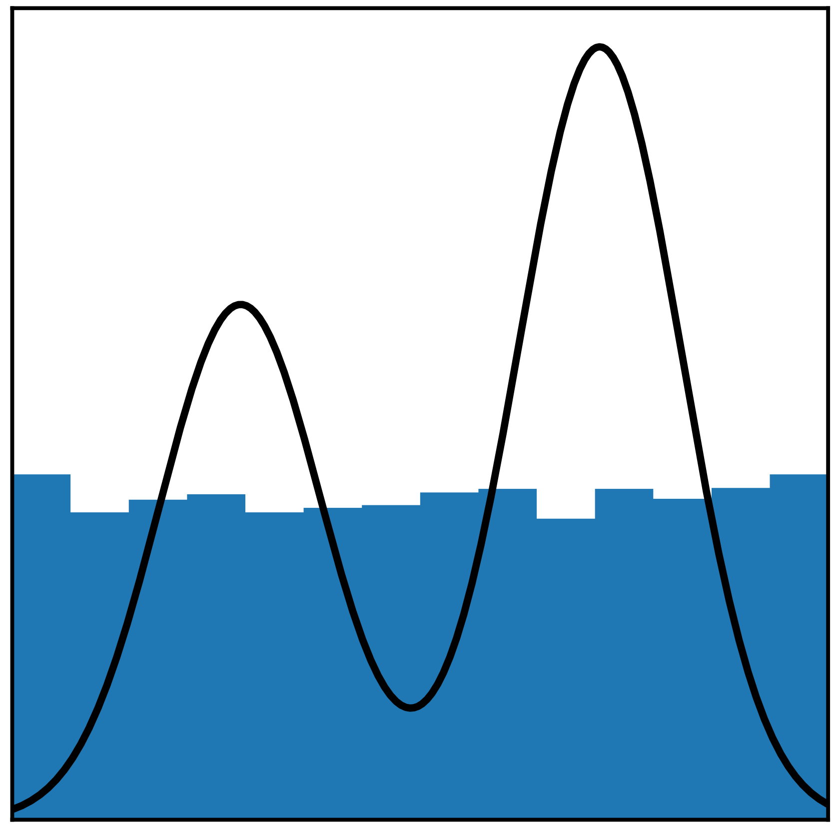


Importance sampling
Find mapping close to integrand

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

Monte Carlo integration + sampling

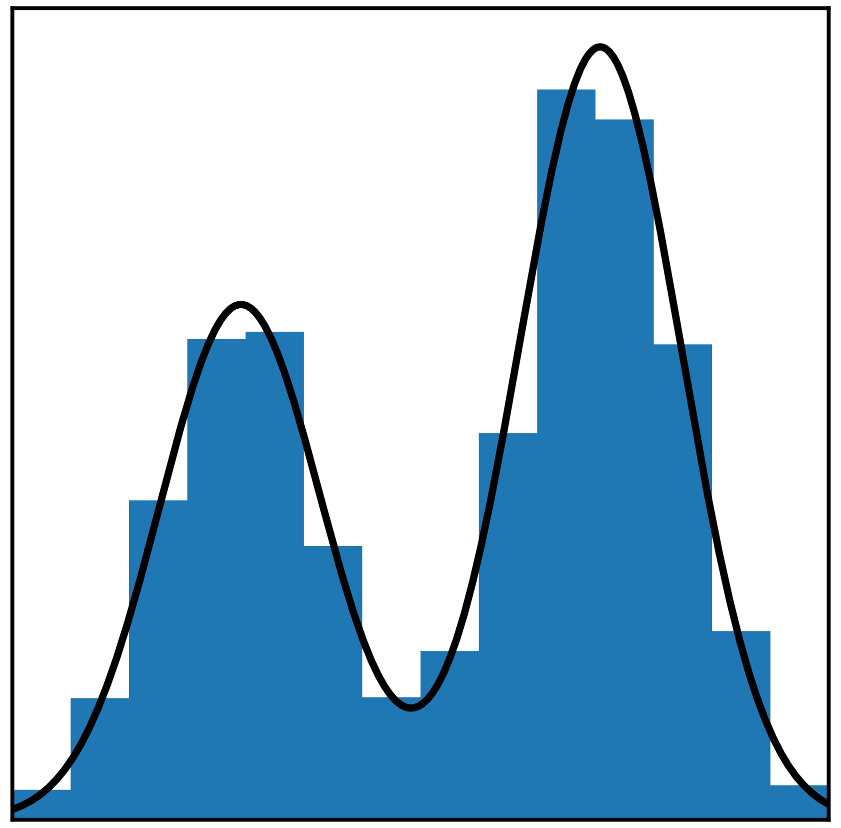
$$I = \int \mathrm{d}x\, f(x)$$



Flat sampling

inefficient

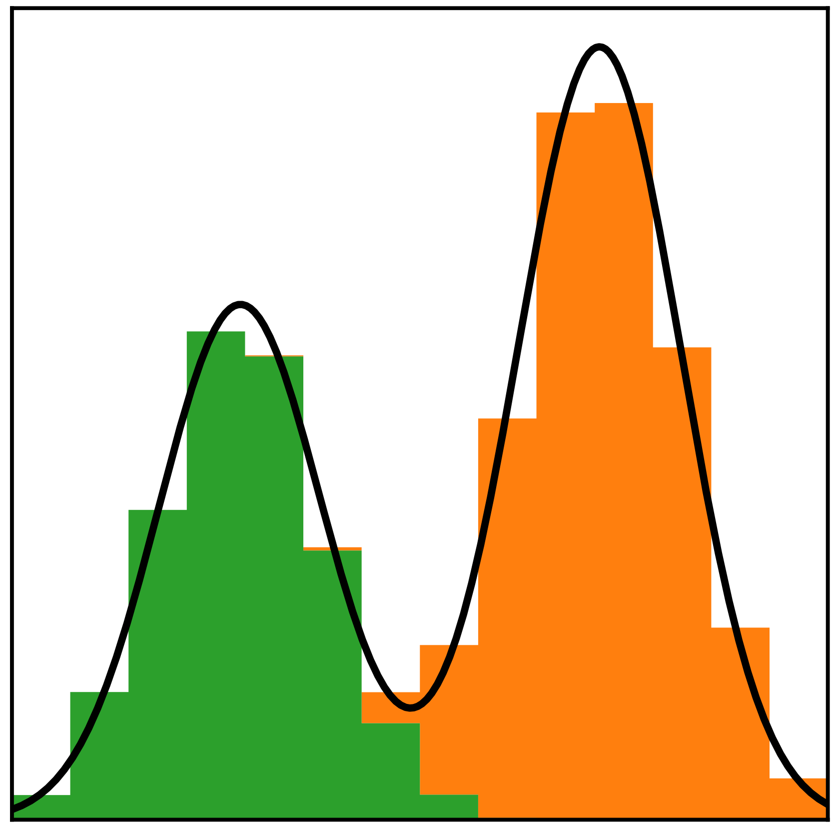
$$I = \langle f(x) \rangle_{x \sim p(x)}$$



Importance sampling

Find mapping close to integrand

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



Multi-channeling

one mapping for each channel

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

Phase space integration

How can we make event generation faster?

Efficient integration and sampling from differential cross section

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

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

Phase space integration

How can we make event generation faster?

Efficient integration and sampling from differential cross section

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

Integrand

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

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

Phase space integration

How can we make event generation faster?

Efficient integration and sampling from differential cross section

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

Sum over channels

MadGraph: build channels from Feynman diagrams

Integrand

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

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

Phase space integration

How can we make event generation faster?

Efficient integration and sampling from differential cross section

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

Sum over channels

MadGraph: build channels from Feynman diagrams

Integrand

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

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

Channel mappings

MadGraph: use propagators, ...
Refine with VEGAS
(factorized, histogram based importance sampling)

Phase space integration

How can we make event generation faster?
Efficient integration and sampling from differential cross section

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

Sum over channels

MadGraph: build channels from Feynman diagrams

Integrand

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

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

Channel weights

MadGraph: $\alpha_i \sim |M_i|^2$
or $\alpha_i \sim \prod |p_k^2 - m_k^2 - \mathrm{i} M_k \Gamma_k|^{-2}$

Channel mappings

MadGraph: use propagators, ...
Refine with VEGAS
(factorized, histogram based importance sampling)

MadNIS: Neural Importance Sampling

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

Use physics knowledge to construct channels and mappings

Normalizing Flow to refine channel mappings

Fully connected network to refine channel weights

Optimize simultaneously with integral variance as loss function

Overview

MadNIS

Basic functionality

- Neural Channel Weights
- Normalizing Flow
- MadGraph matrix elements
- MadEvent channel mappings

Improved Multichanneling

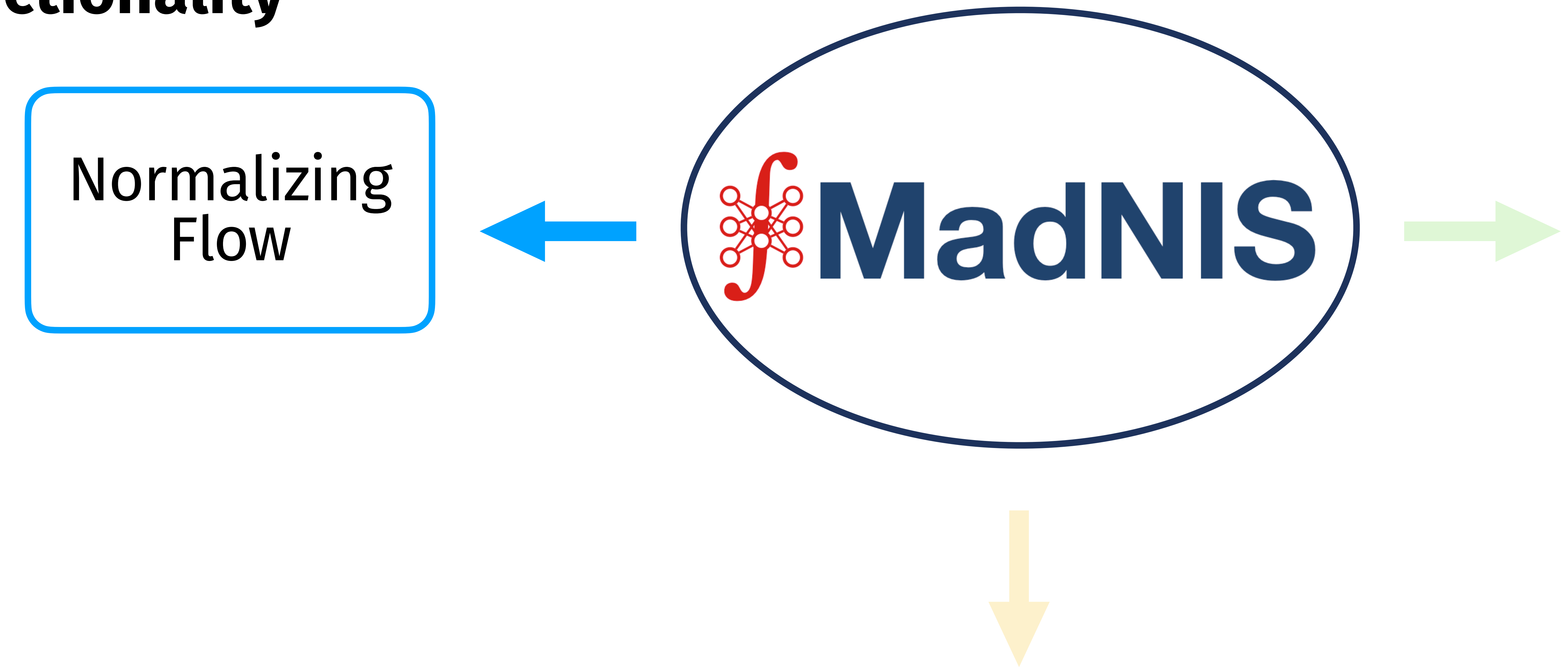
- Stratified sampling/ training
- Symmetries between channels
- Removing channels
- Partial weight buffering

Improved training

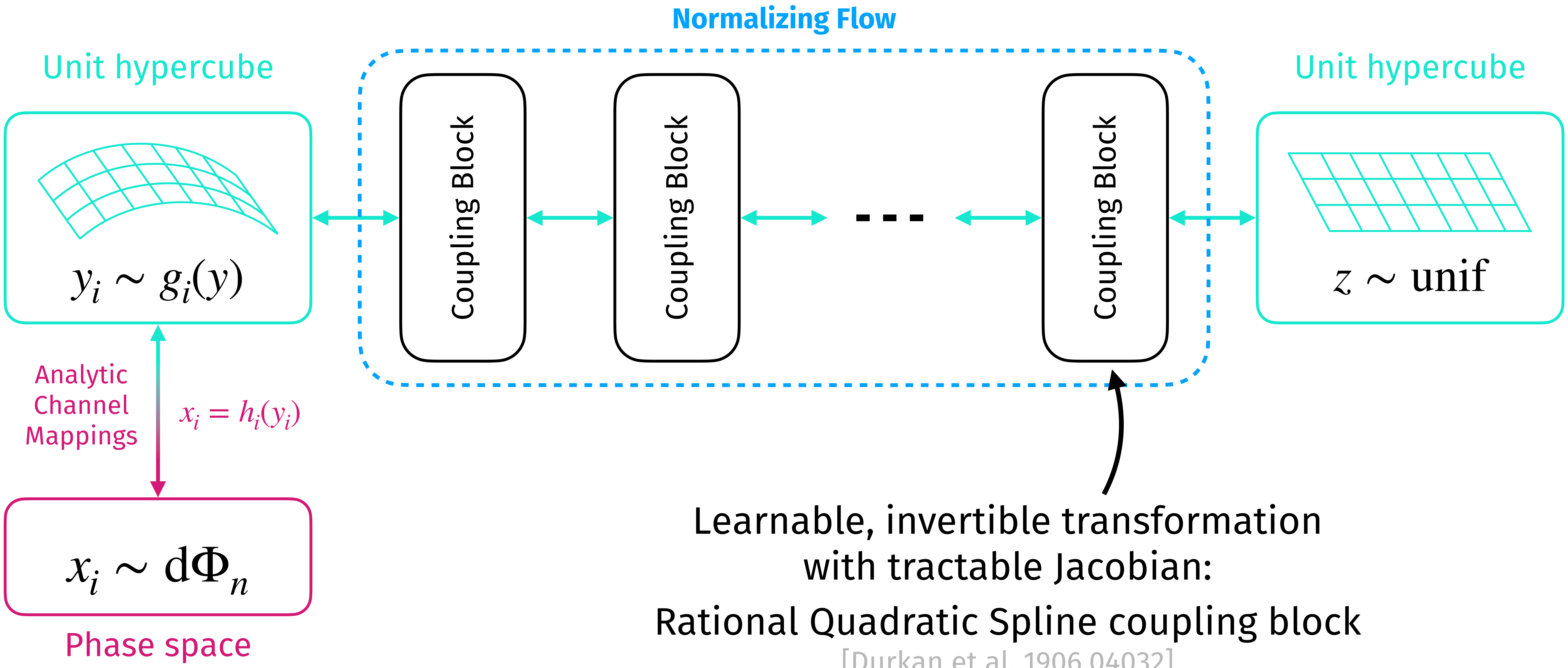
- VEGAS initialization
- Buffered training
- Surrogate integrand

Overview

Basic functionality



Neural Importance Sampling



Learnable, invertible transformation with tractable Jacobian:

Rational Quadratic Spline coupling block

[Durkan et al, 1906.04032]

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

Overview

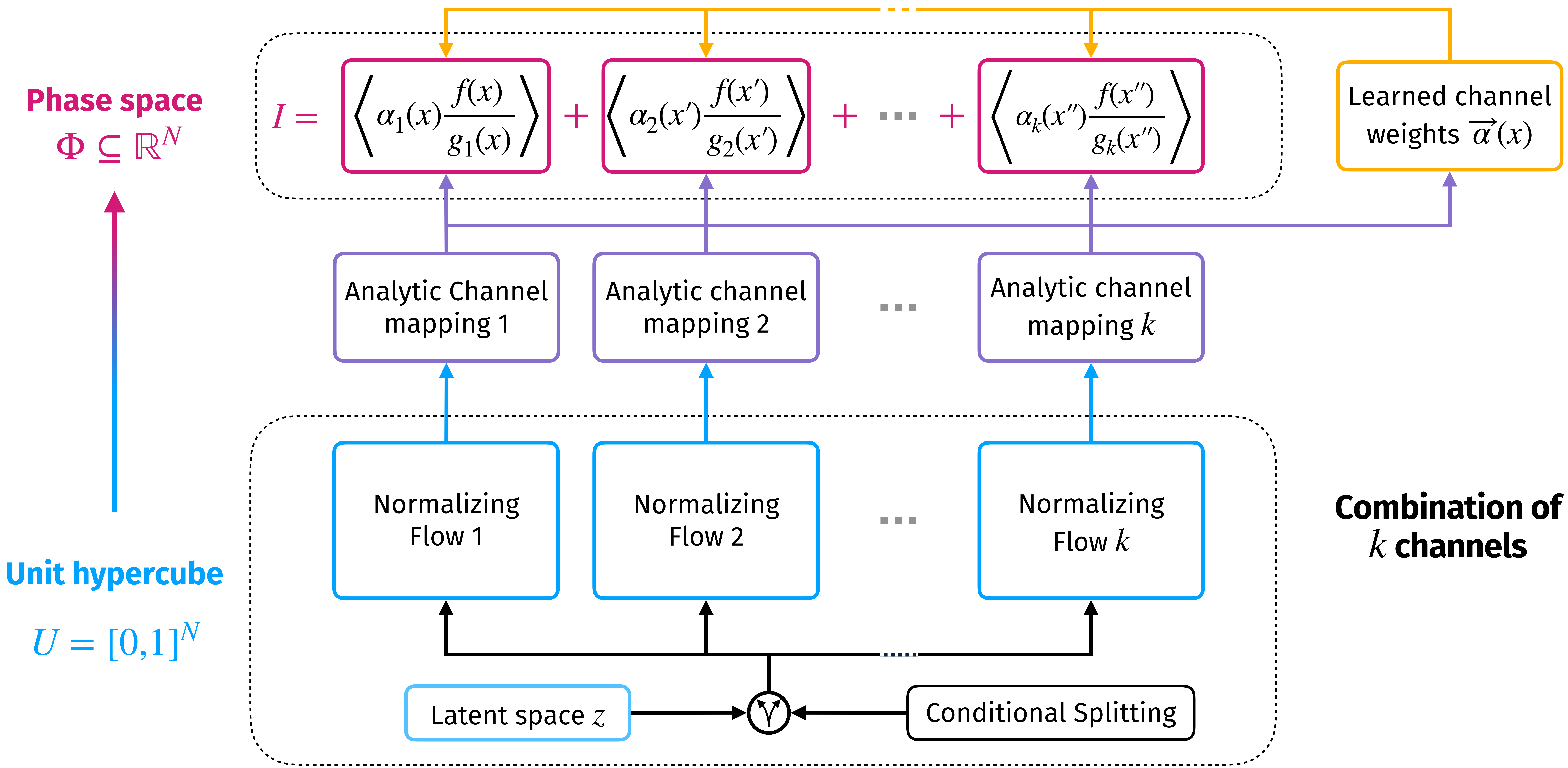
Basic functionality

Neural Channel Weights

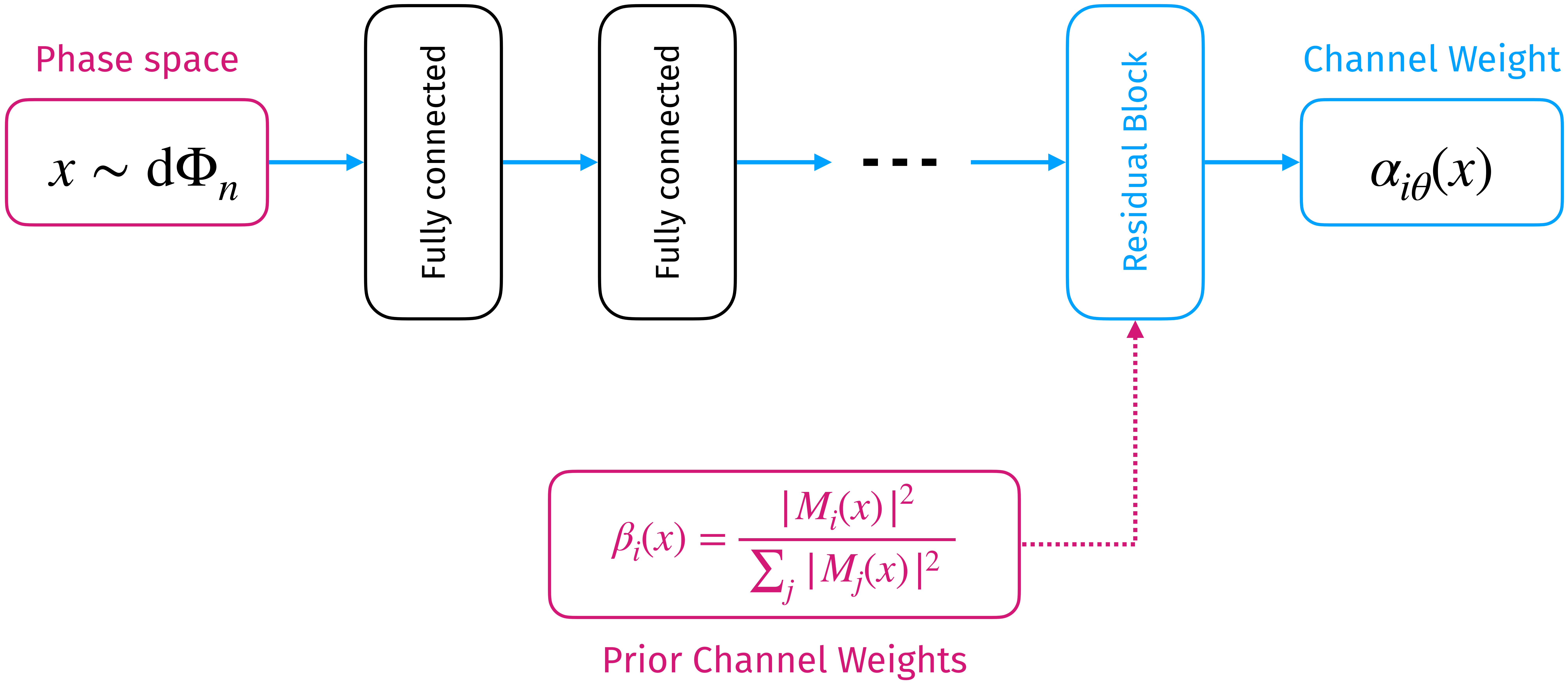
Normalizing Flow

MadNIS

MadNIS: Neural Importance Sampling



Neural Channel Weights



Loss function

Training objective:
Minimize total variance

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

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

Optimal MC weights depend on N_i

$\downarrow$

assume choice of N_i during training:
use stratified sampling

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

MadNIS loss function

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

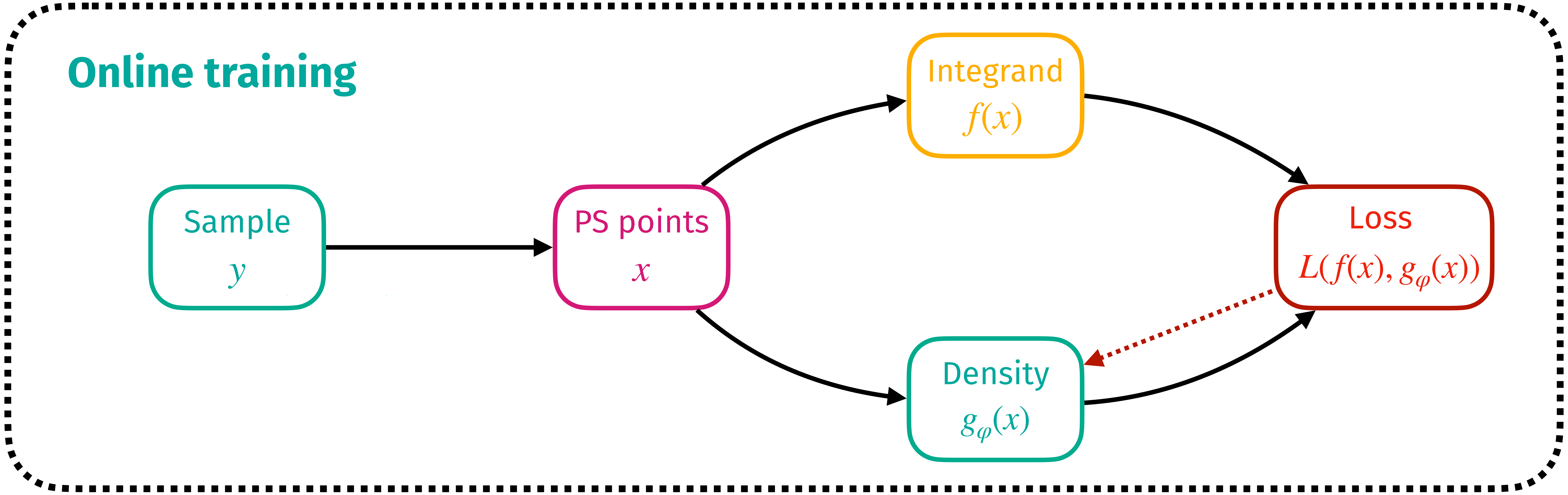
Overview

MadNIS

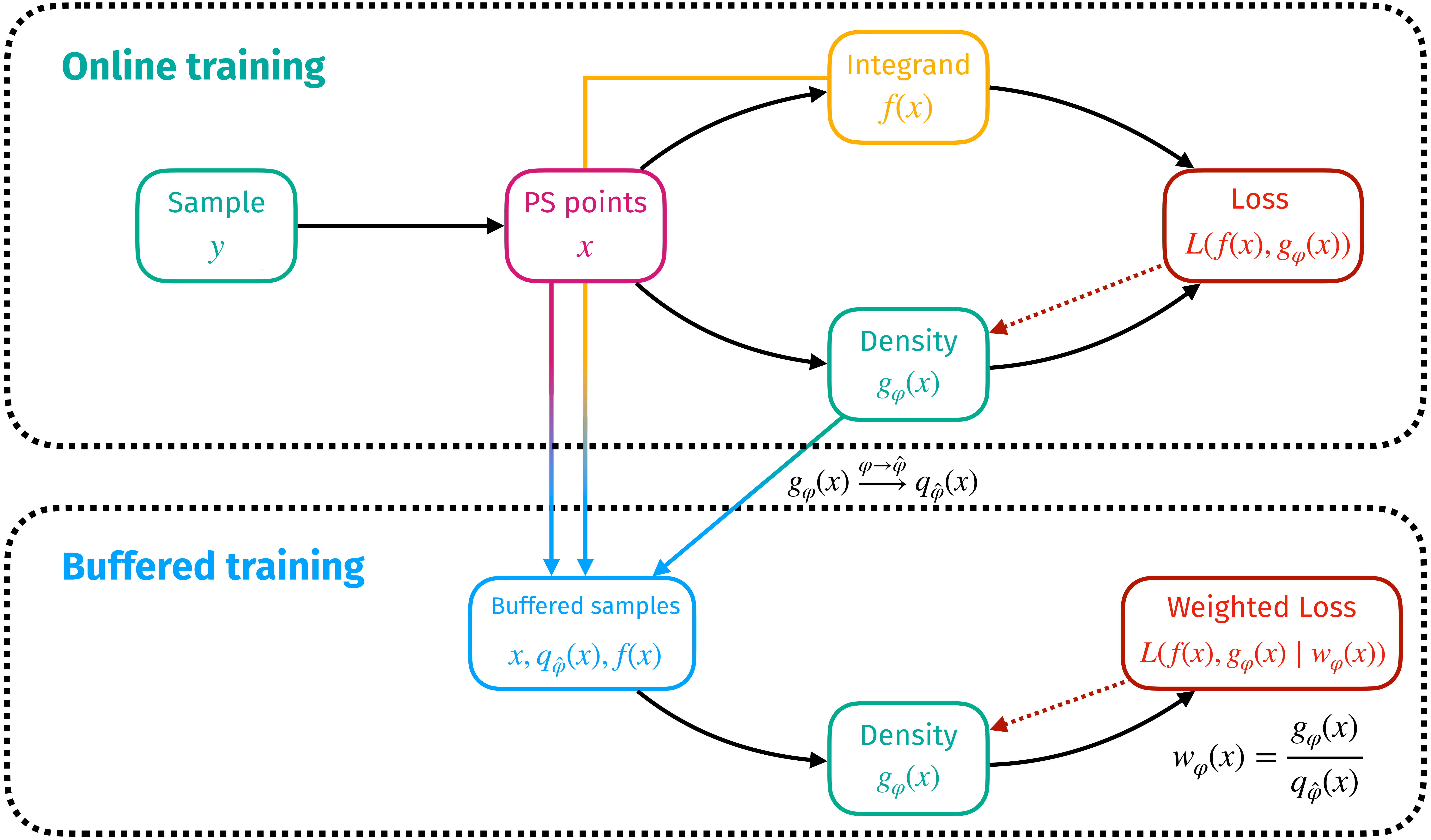
Improved training

Buffered training

Buffered Training



Buffered Training



Buffered Training

Training algorithm

generate new samples, train on them, save samples

↓

train on saved samples n times

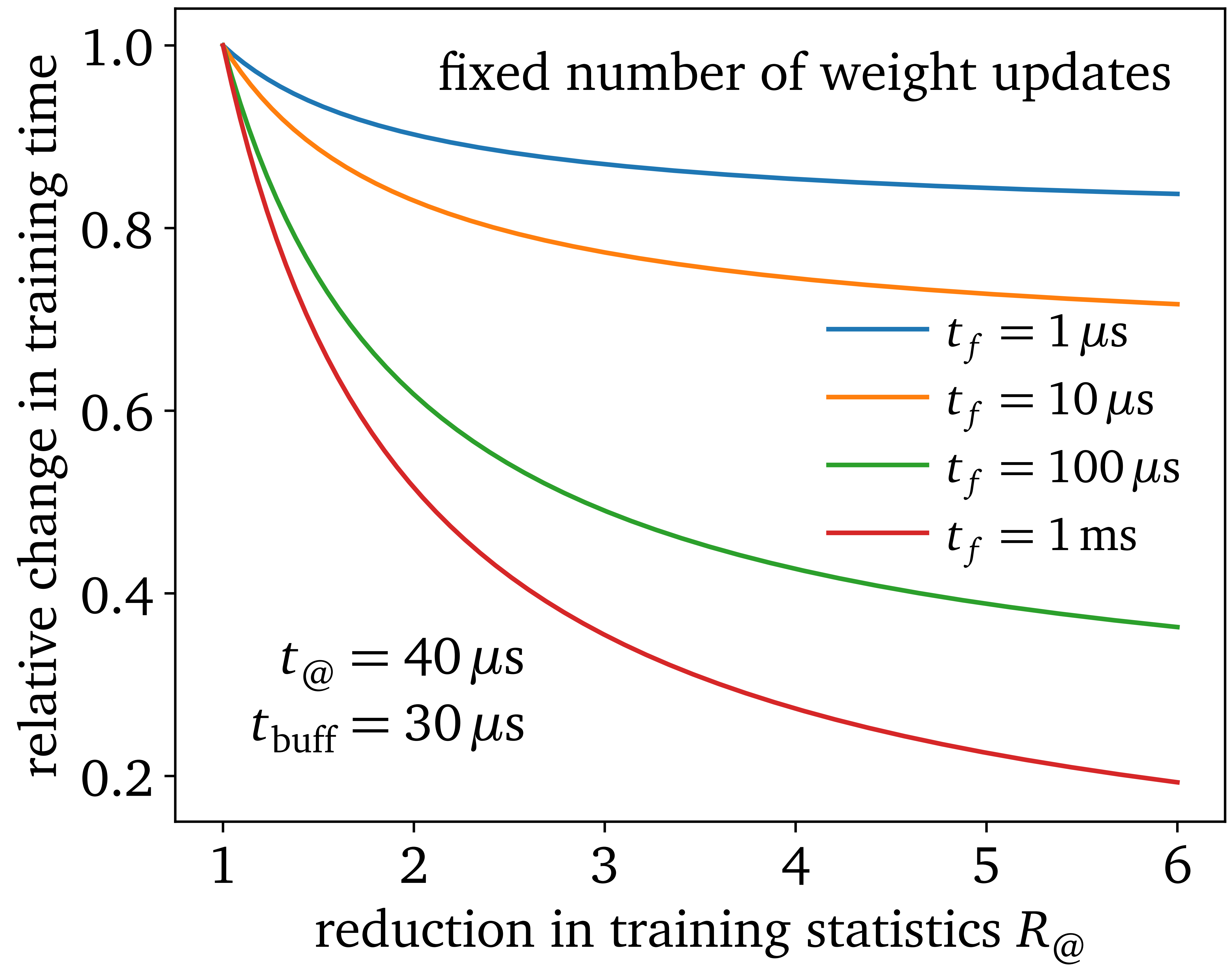
↓

repeat

↓

Reduction in training statistics by

$$R_@ = n + 1$$



Overview

Improved training

VEGAS initialization

VEGAS Initialization

	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

Combine advantages:

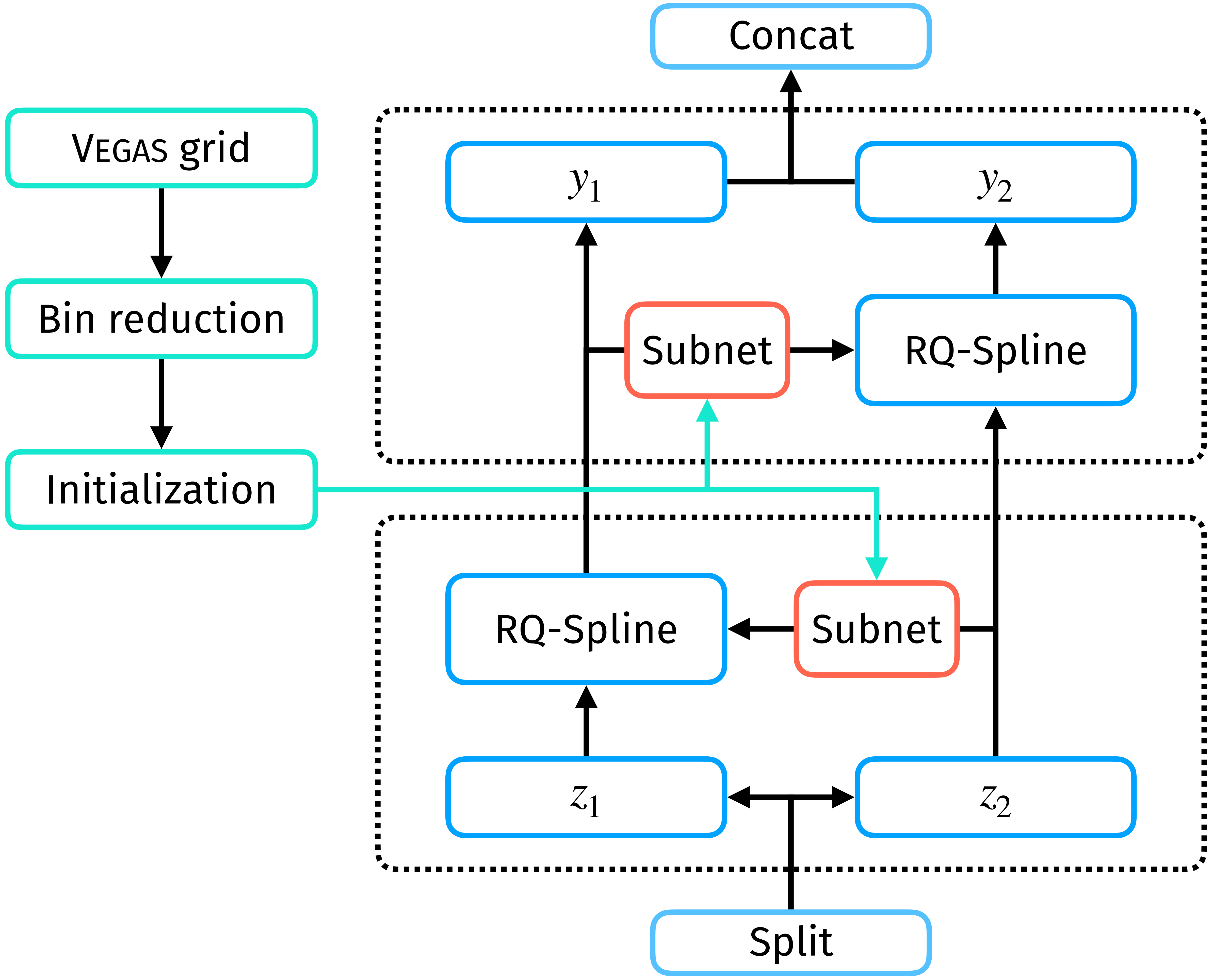
Pre-trained VEGAS grid as starting point for flow training

VEGAS Initialization

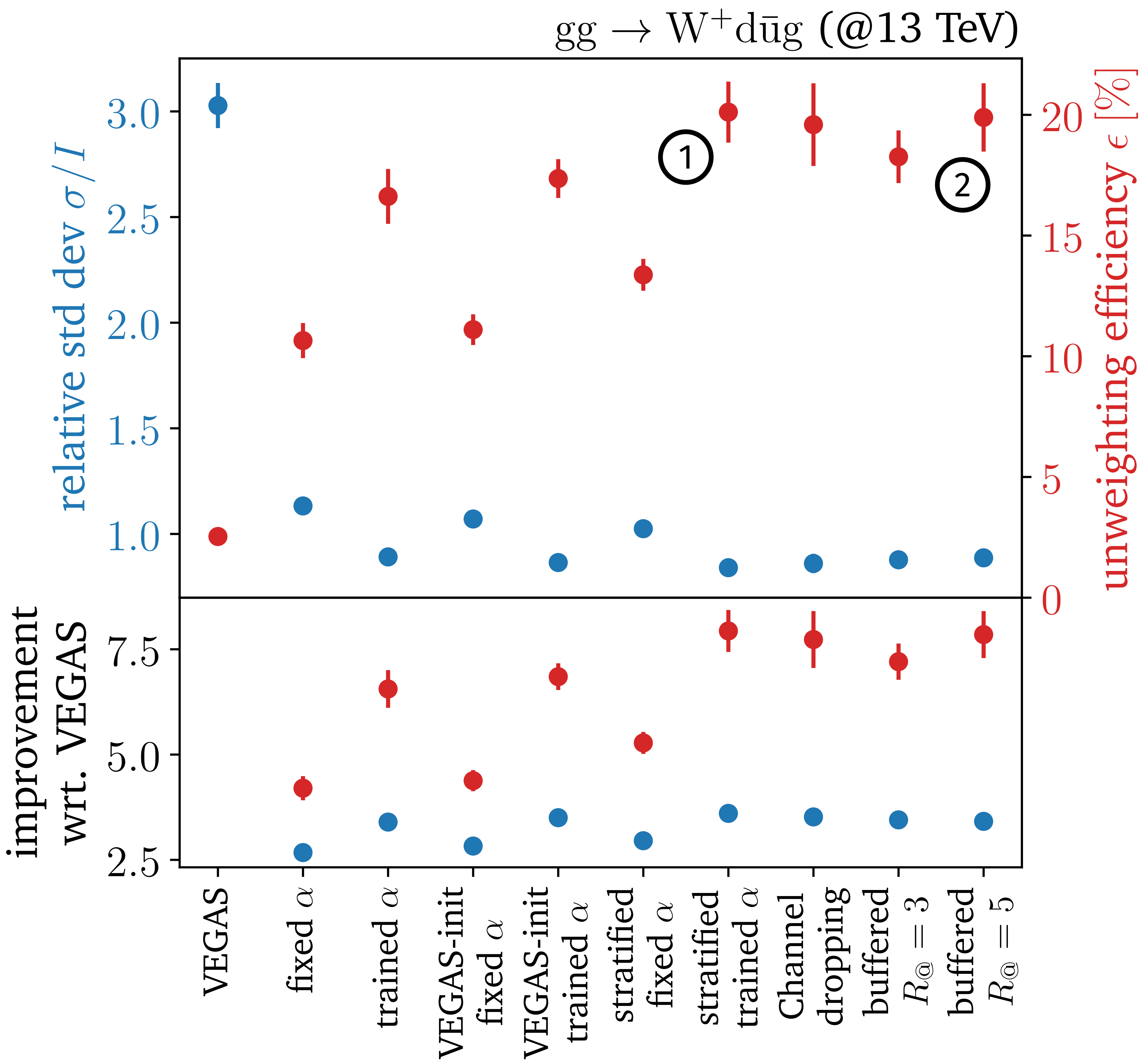
	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

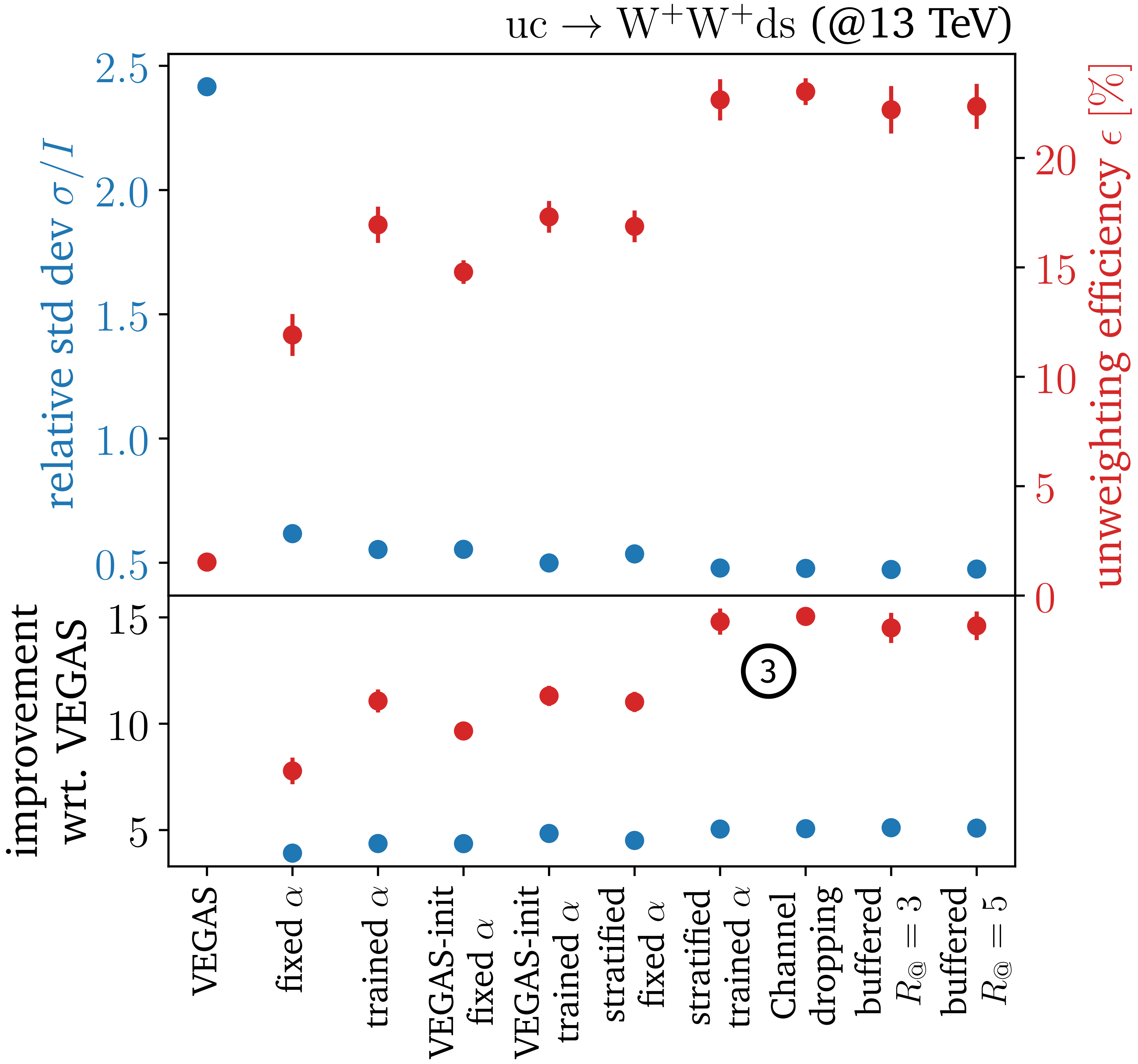
Combine advantages:

Pre-trained VEGAS grid as starting point for flow training



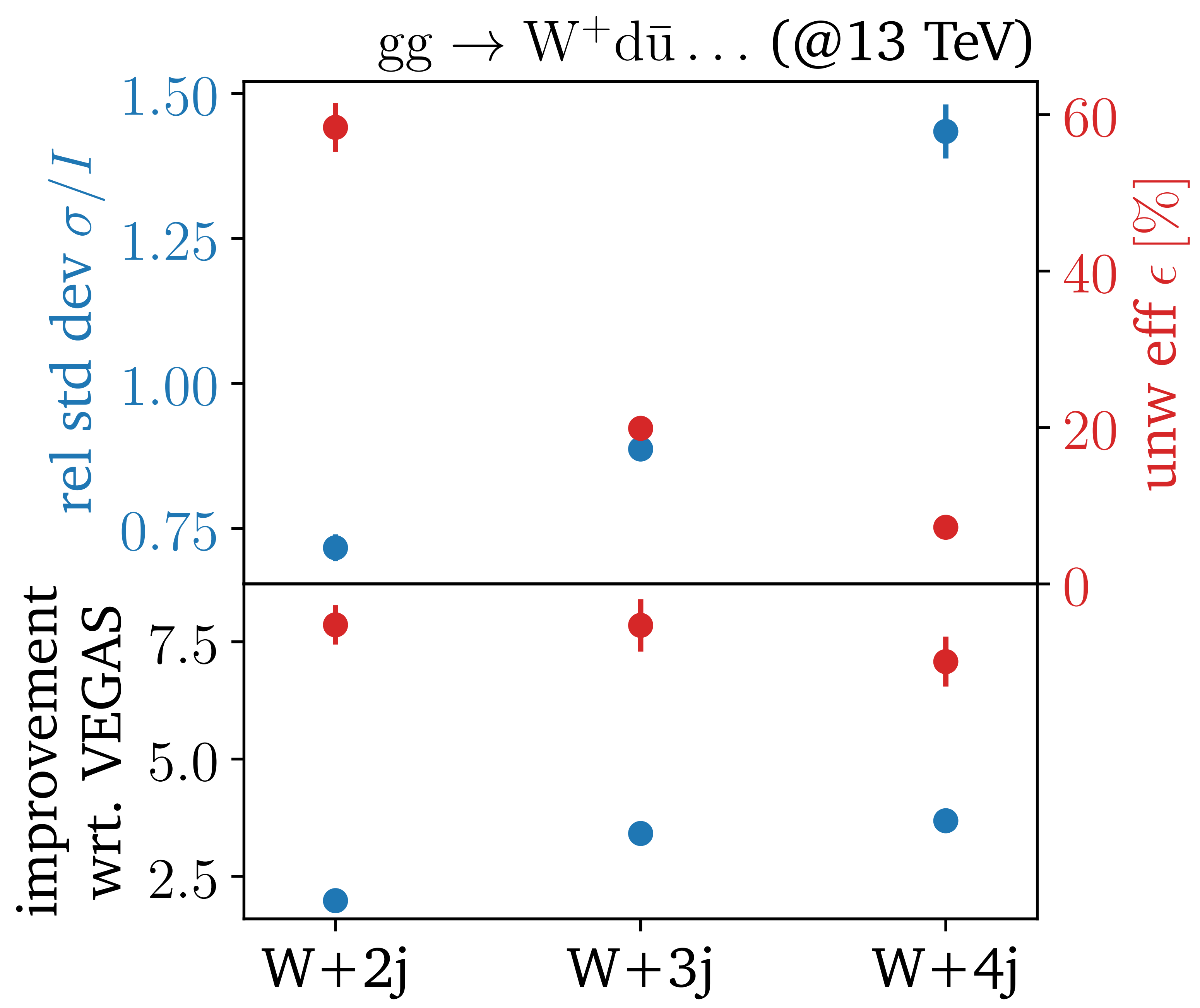
LHC processes

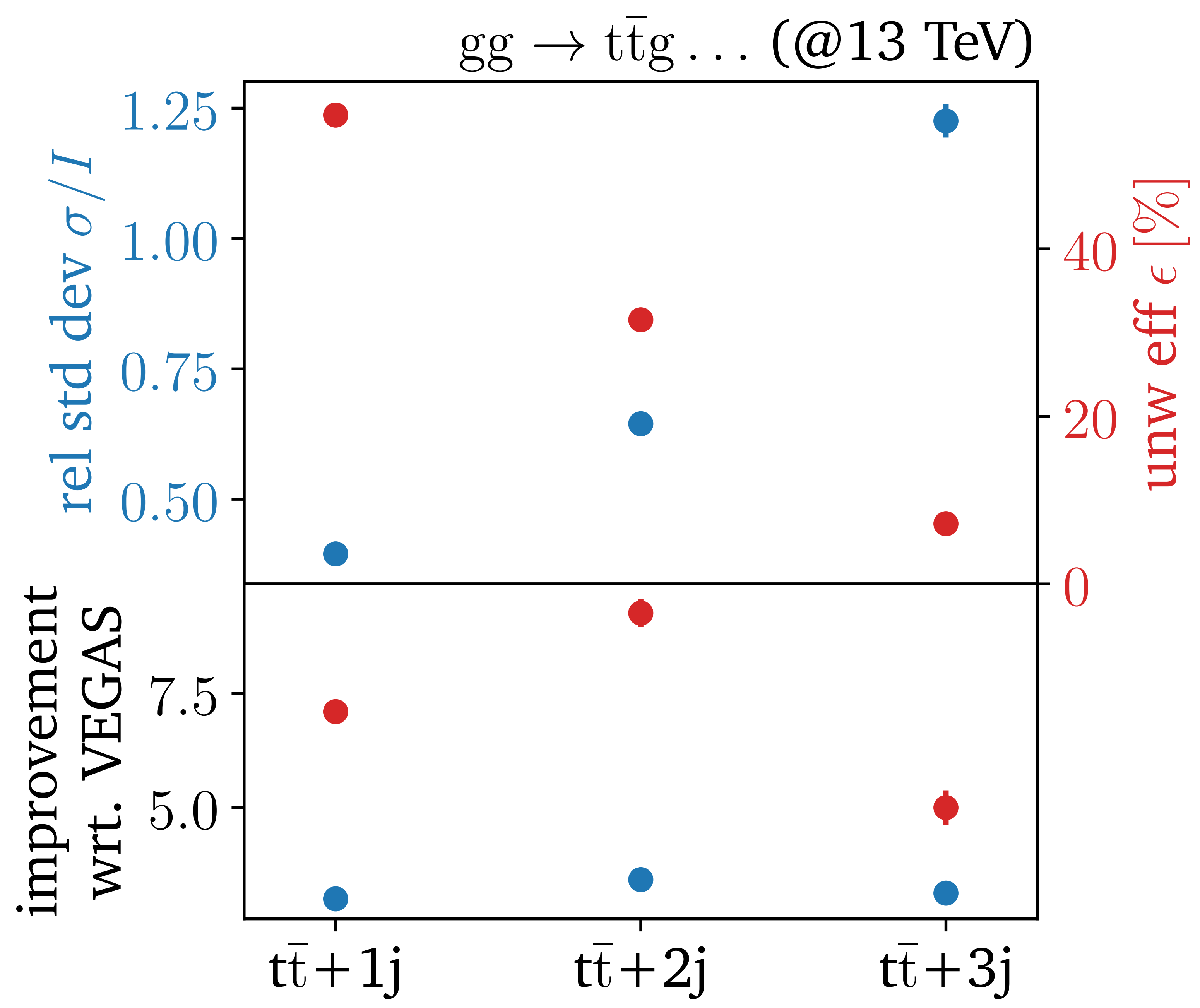




1. Excellent results by combining all improvements!
2. Same performance with buffered training
3. Even larger improvements for process with large interference terms

Scaling with multiplicity



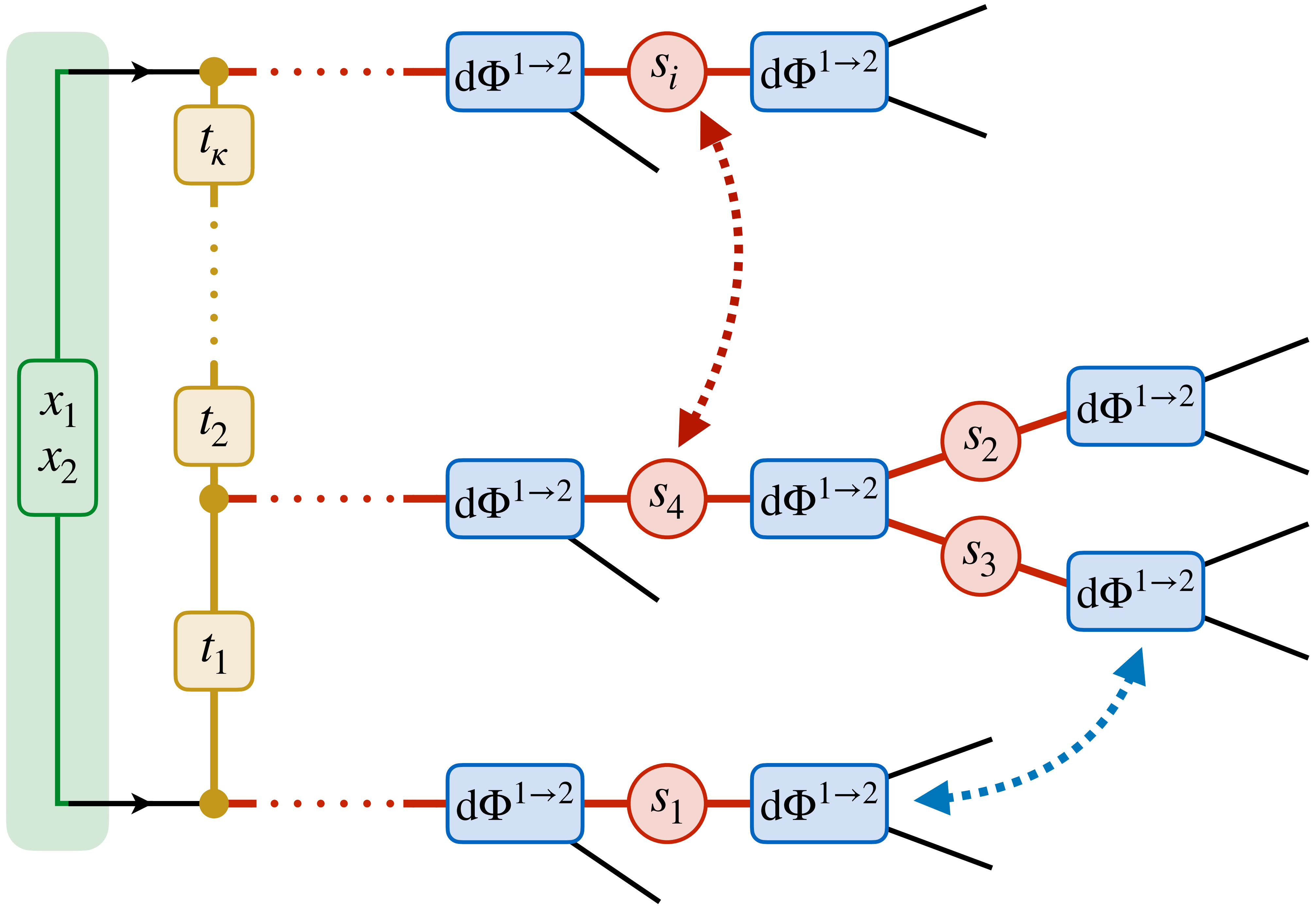


$gg \to W^+ d\bar{u}gg$
384 channels, 108 symm.
7x better than VEGAS

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

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

Differentiable MadNIS-Lite



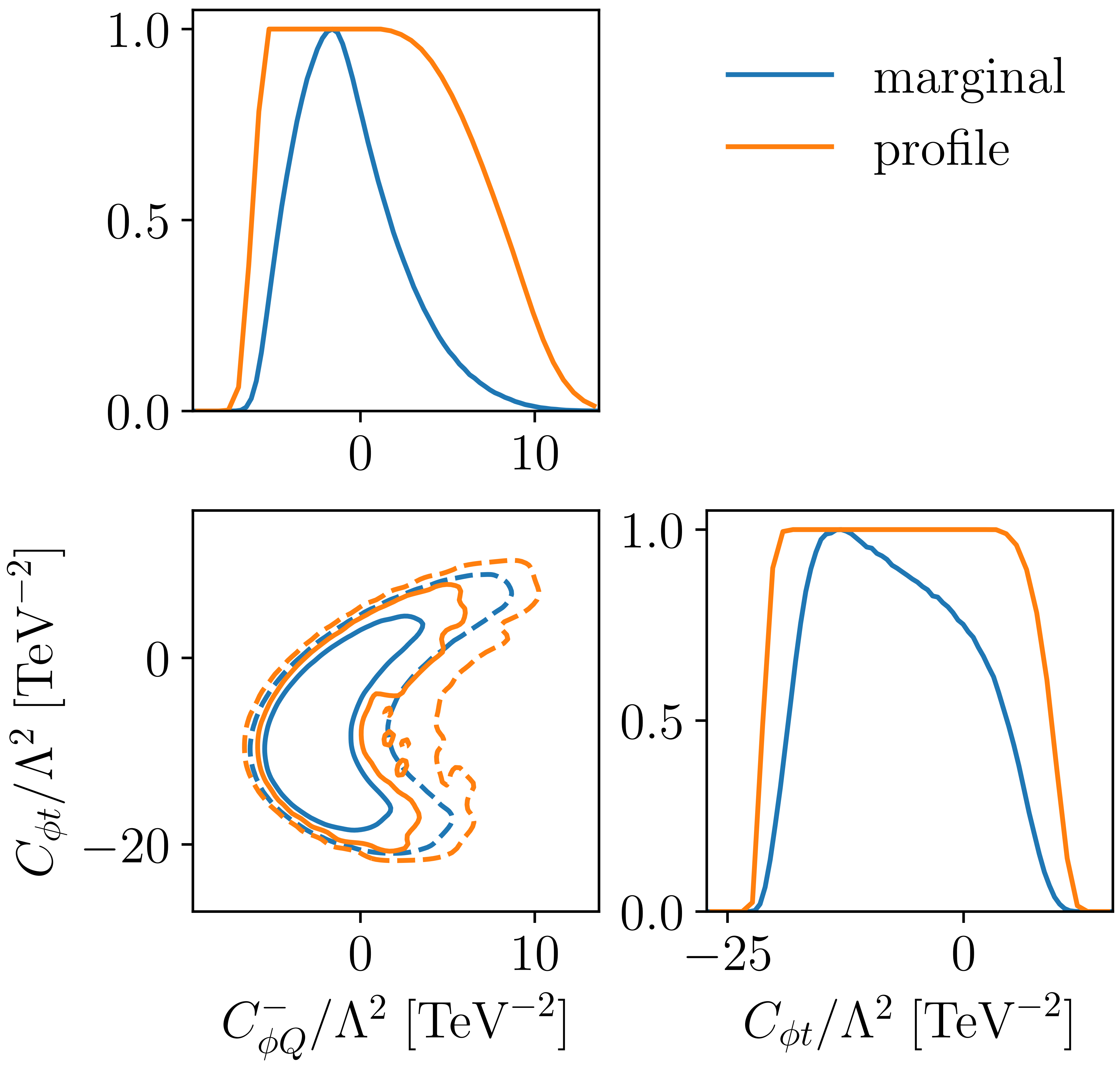
Differentiable MadNIS-Lite [2408.01486]
TH, Olivier Mattelaer, Tilman Plehn, Ramon Winterhalder

- Build PS mappings from Feynman diagrams
  → implemented in PyTorch
  → **Fully differentiable** and **invertible**

- Build in small trainable components, with parameters shared between
  → all components of same type
  → all channels

- Learn physics of PS mappings
  → **interpretability**
  → **train on n jets, generate n+1 jets**

MadNIS technology for SFitter



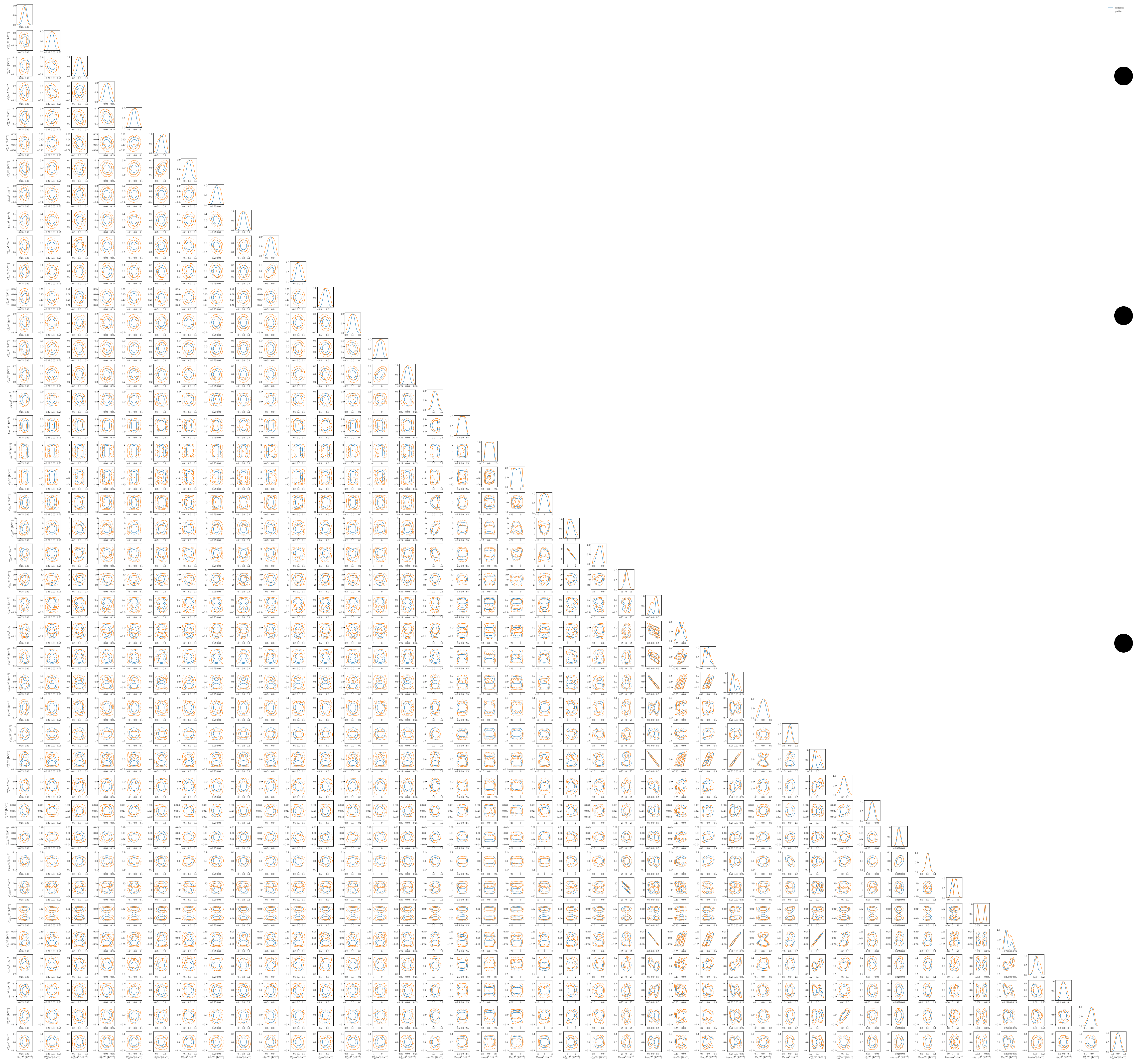
- Apply neural importance sampling to SFitter likelihood

- Combined SMEFT fit in Higgs and Top sector
  → 42 Wilson coefficients
  → ~500 datapoints from various analyses

- **Efficient profiling and marginalization**
  → before: days on CPU cluster
  → now: **a few hours on a single GPU**

Profile likelihood on ML steroids [2411.00942]
TH, Tilman Plehn, Nikita Schmal

MadNIS technology for SFitter



- Apply neural importance sampling to SFitter likelihood
- Combined SMEFT fit in Higgs and Top sector
  → 42 Wilson coefficients
  → ~500 datapoints from various analyses
- **Efficient profiling and marginalization**
  → before: days on CPU cluster
  → now: **a few hours on a single GPU**

Profile likelihood on ML steroids [2411.00942]
TH, Tilman Plehn, Nikita Schmal

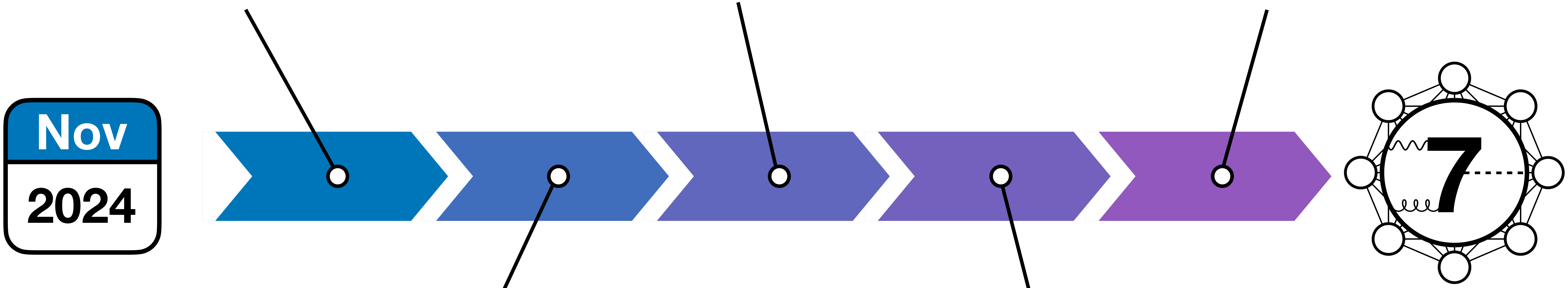
Outlook

Release of MadNIS package
- python library
- easy install with 'pip install'

→ **December 2024**

New MadEvent7
- fully vectorized mappings
- multiple backends (c++, cuda, python,…)

Release of MadGraph7
- rigorous testing
- reliable default settings dep. on hardware/process/…



Fully integrate into MG5aMC
- multiple partonic processes
- optimized API
- merge with MG@GPU

MadNIS@NLO
- subtraction-aware sampling
- fast ML amplitudes (NLO)

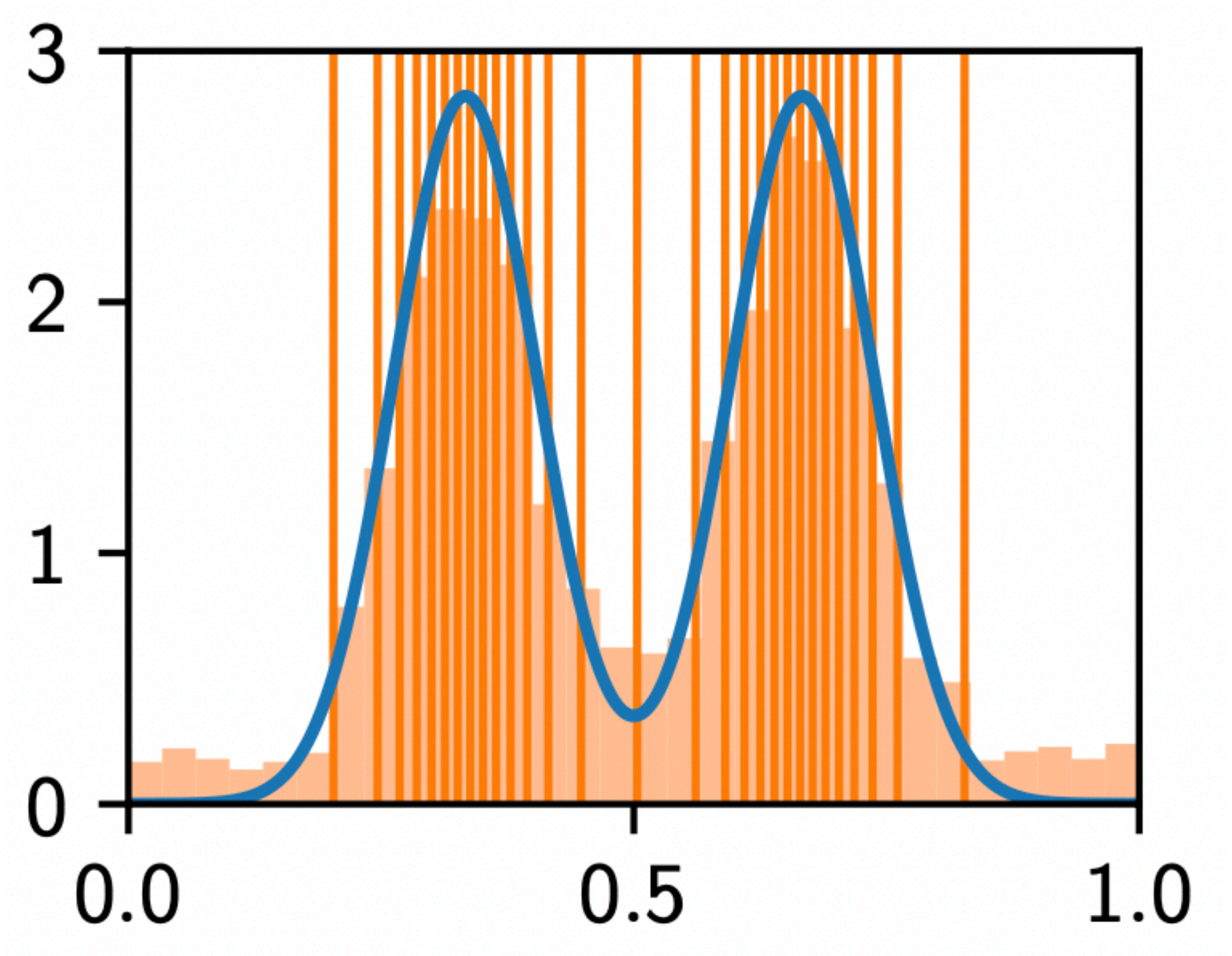
Appendix

VEGAS algorithm

Factorize probability

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

Fit bins with equal probability
and varying width

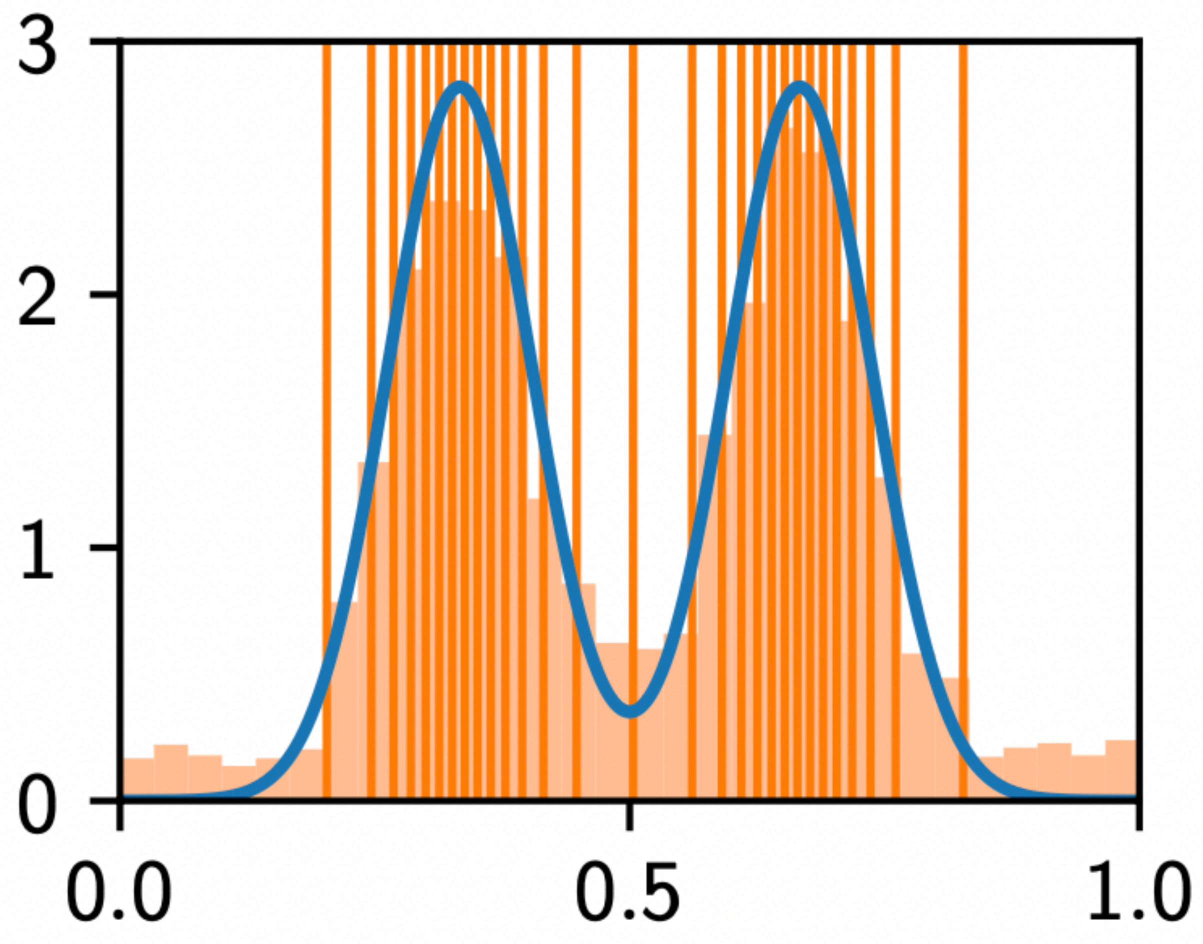


[G. P. Lepage, 1978]

VEGAS algorithm

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

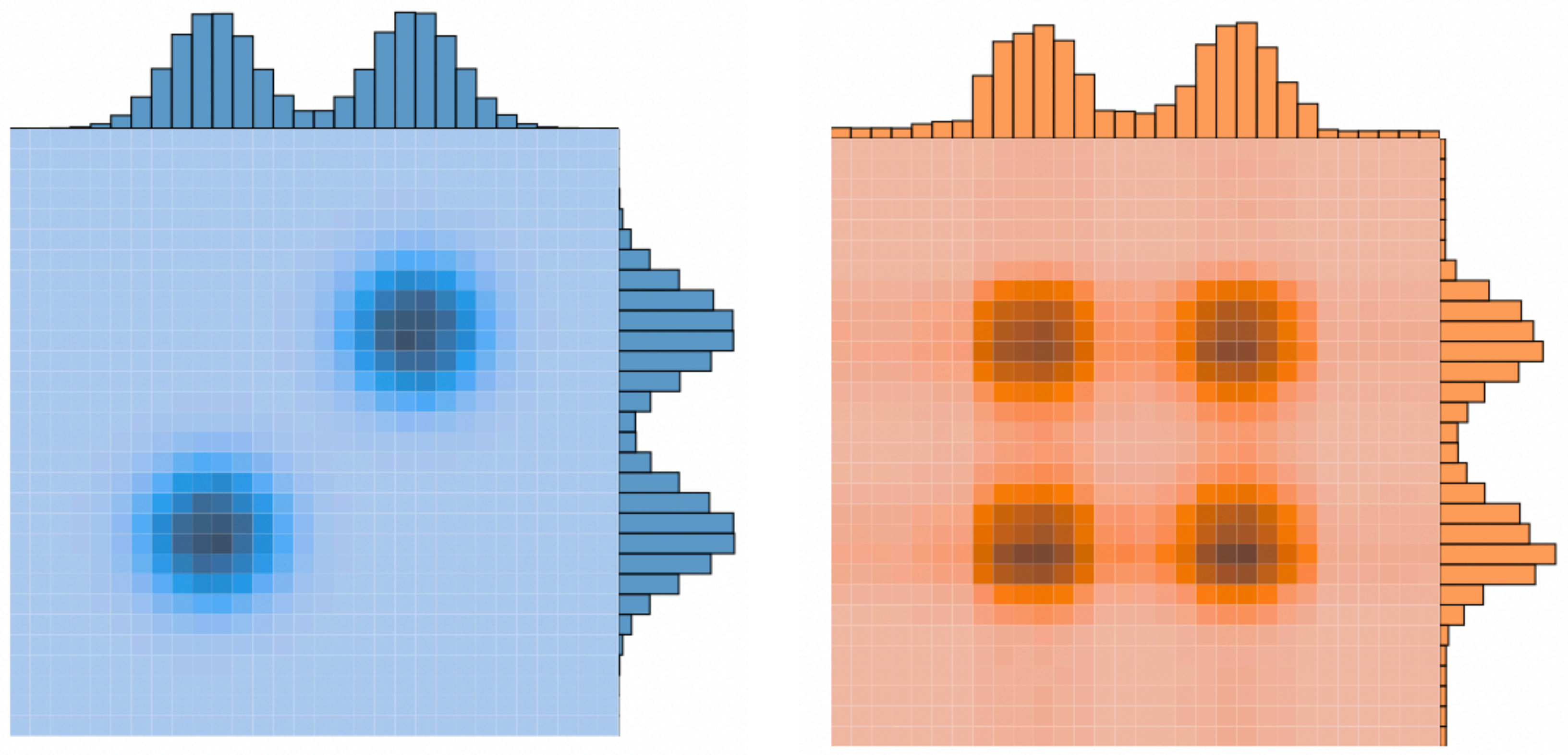
Fit bins with equal probability
and varying width



⊕ Computationally cheap

⊖ High-dim and rich peaking functions
→ slow convergence

⊖ Peaks not aligned with grid axes
→ phantom peaks

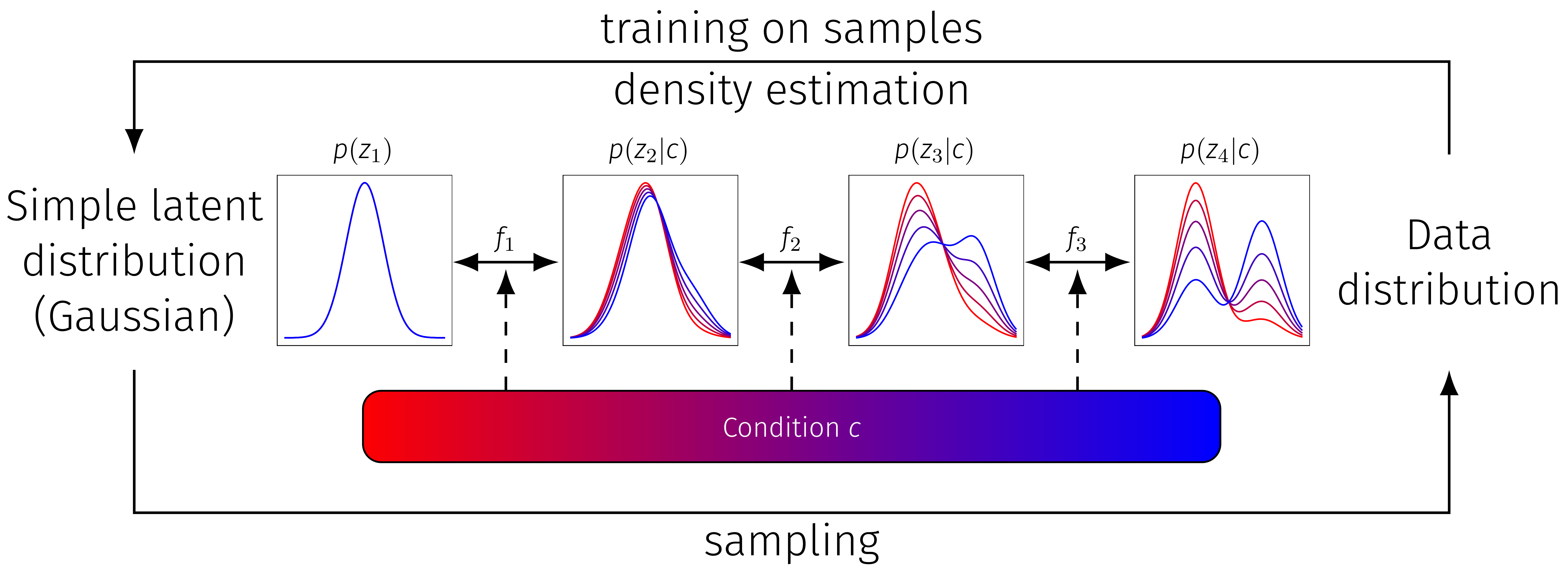


[G. P. Lepage, 1978]

Normalizing Flows

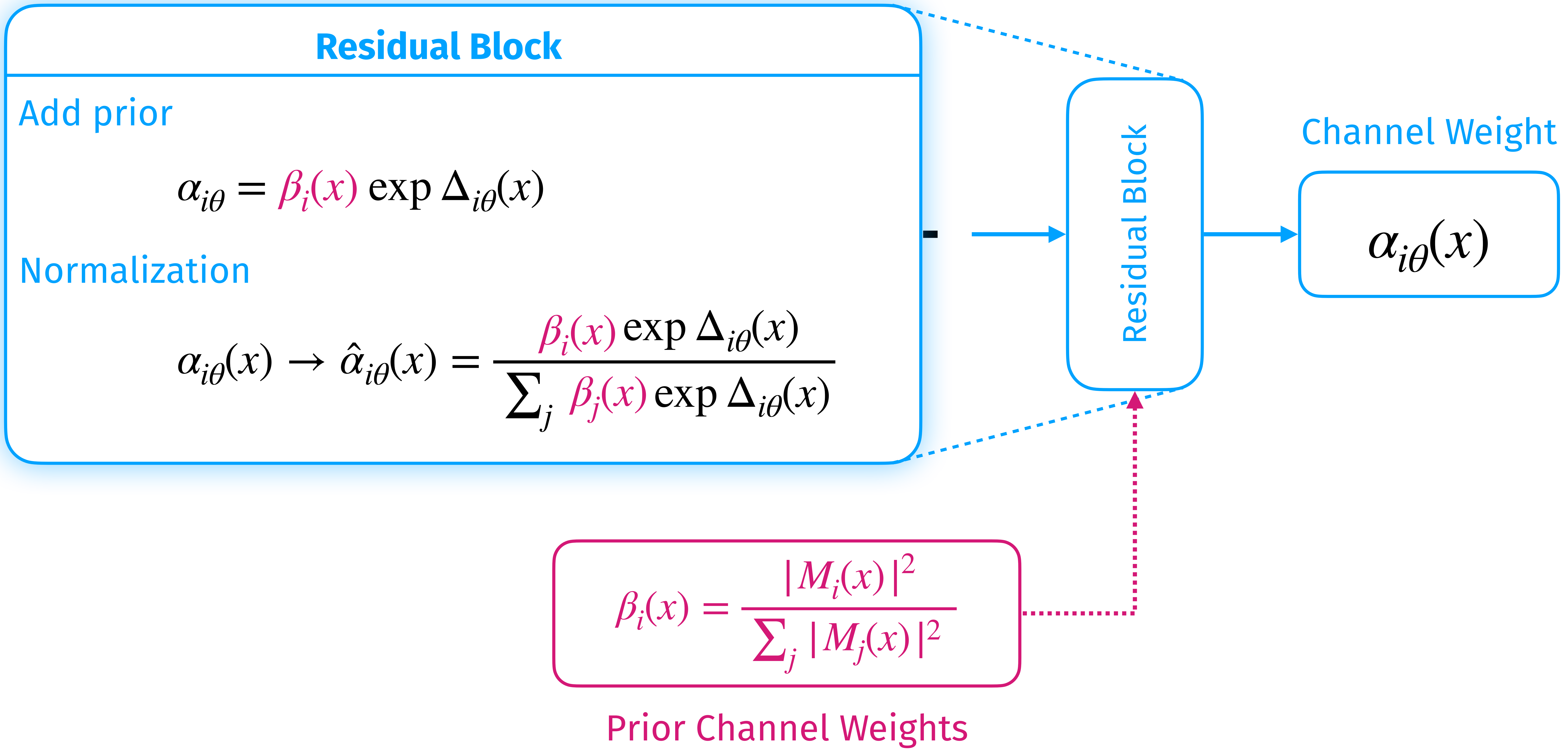
Chain of **invertible**, **learnable** transformations with **exact likelihood** from change of variables formula

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



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

Neural Channel Weights



Improved Multichanneling

Use symmetries

Groups of channels only **differ by permutations** of final state momenta

↓

use **common flows** and combine in loss function

Stratified training

Channels have different contributions to the total variance

↓

more samples for channels with **higher variance** during training

Channel dropping

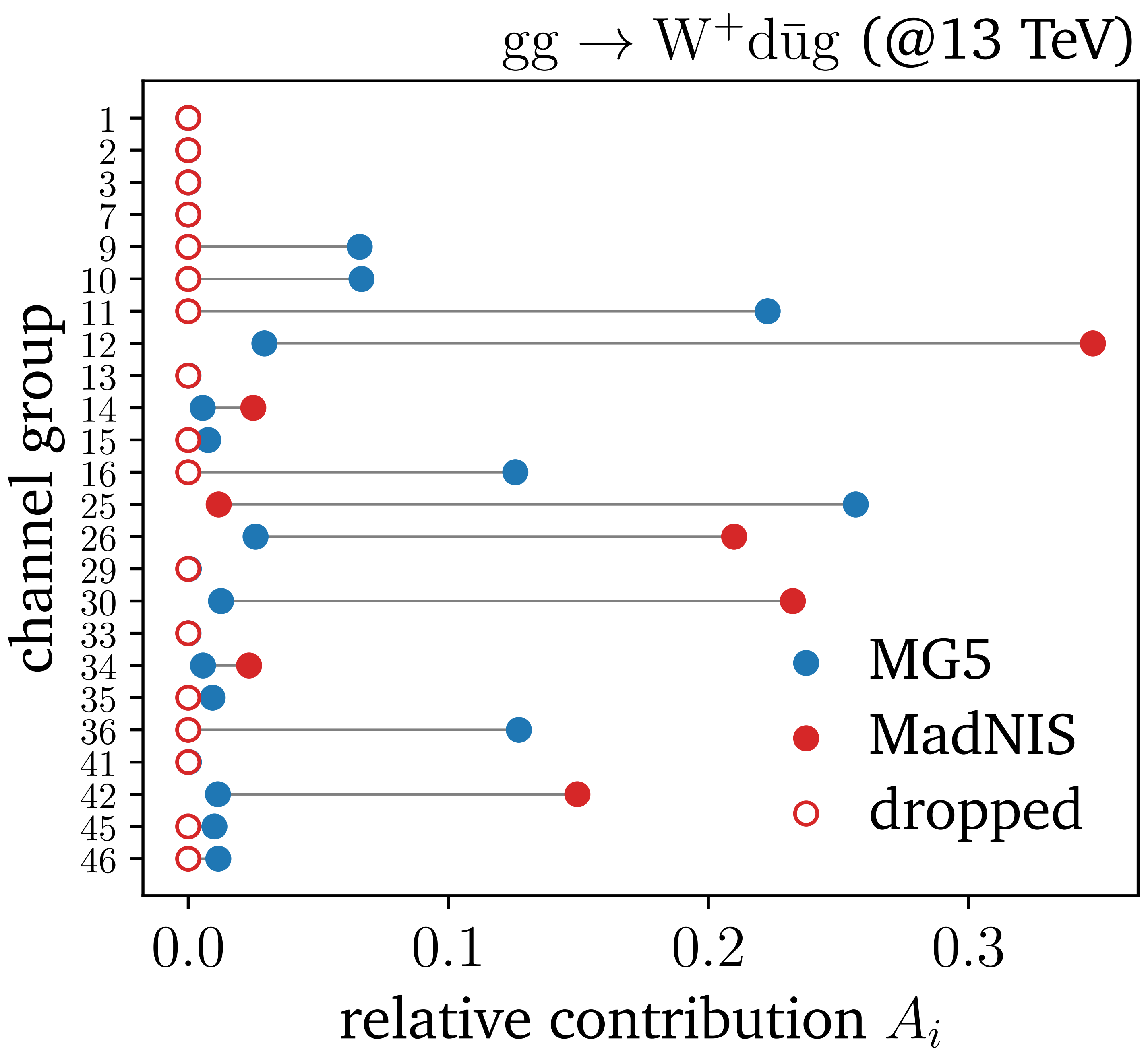
MadNIS often **reduces contribution** of some channels to total integral

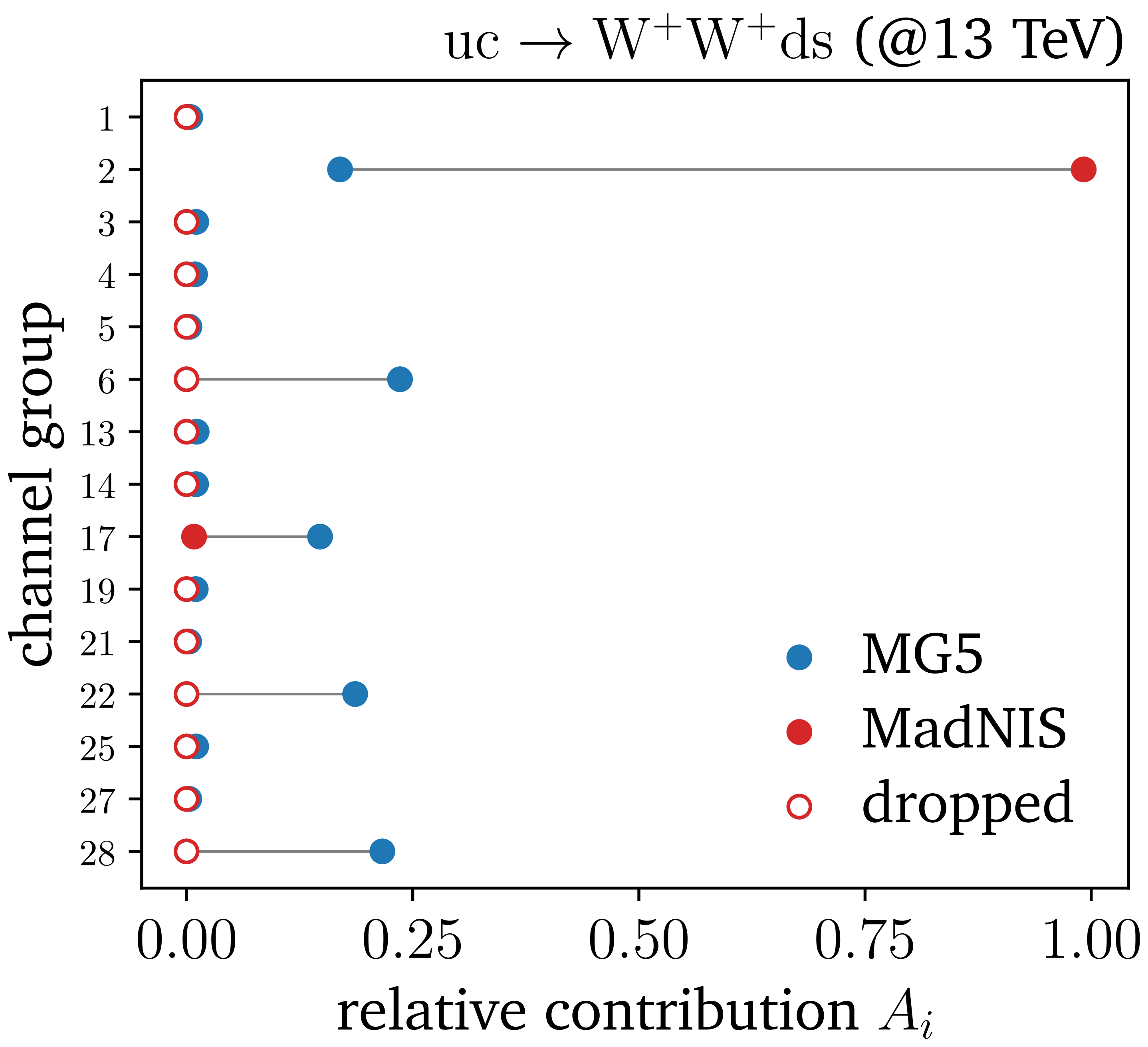
↓

remove insignificant channels from the training completely

Reduced complexity
Improved stability

Learned channel weights





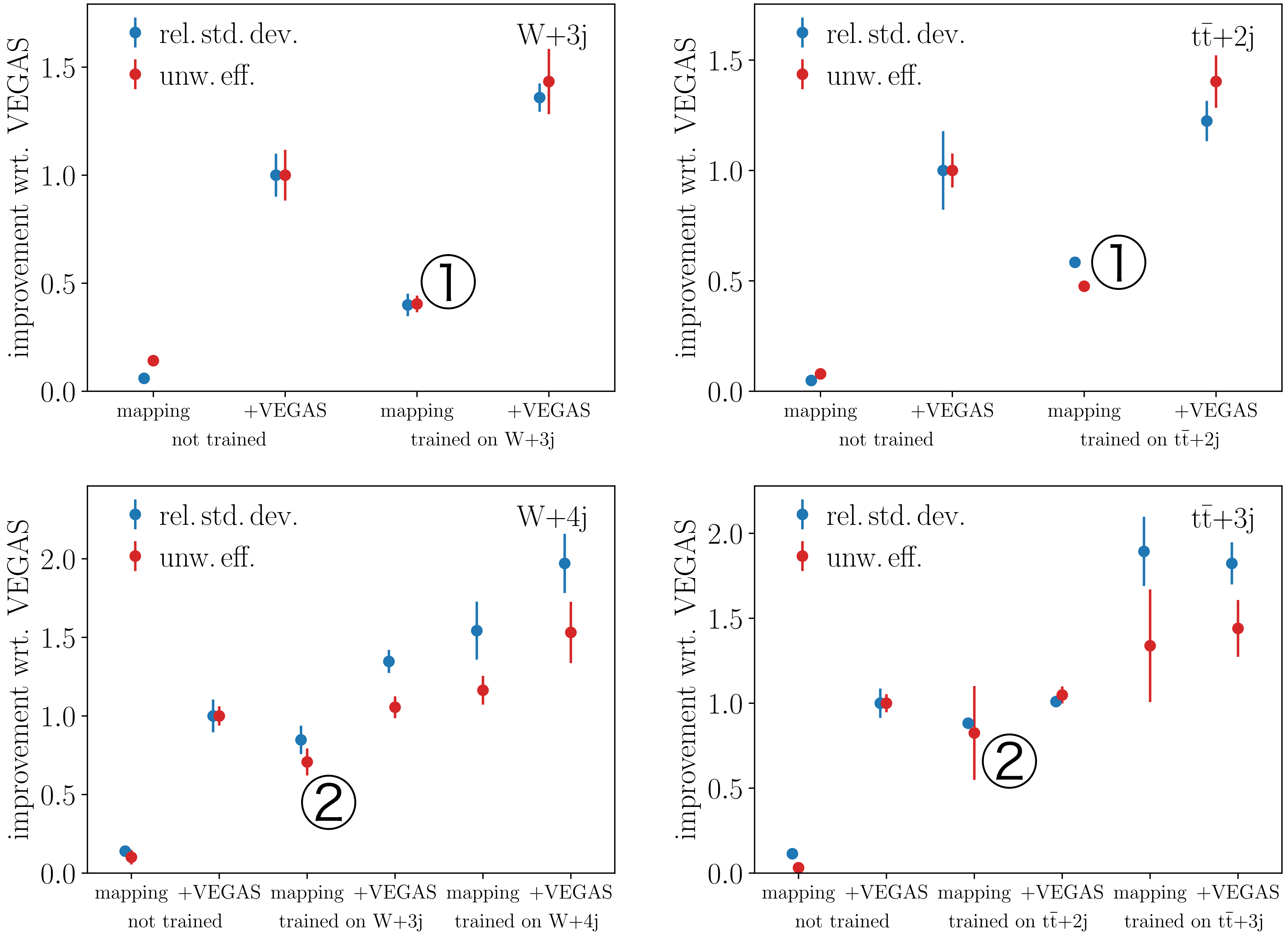
MadNIS often sends weight of many channels to 0

↓

dropping channels makes training and
event generation more stable and efficient

Performance

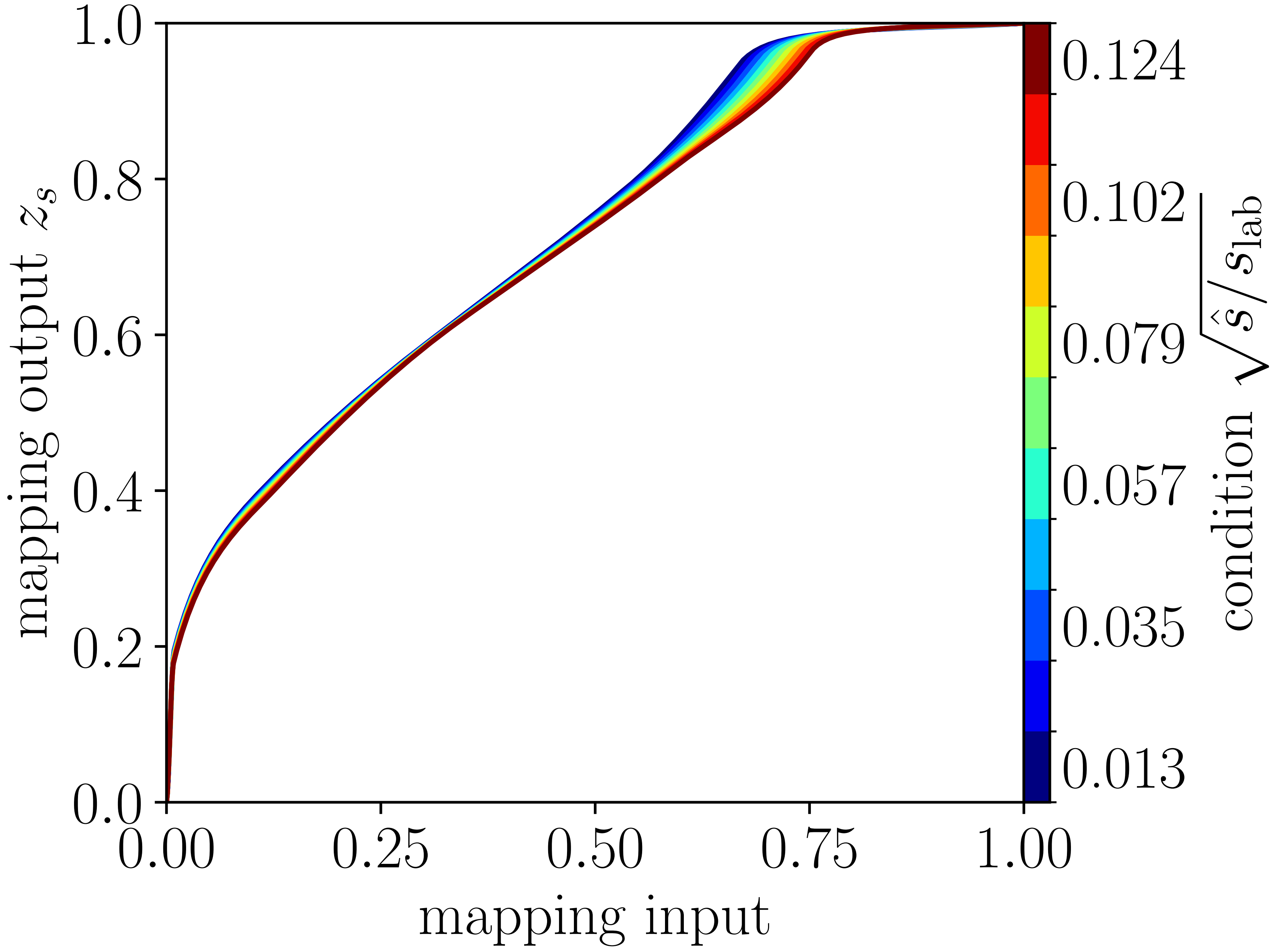
W+jets | $t\bar{t}$+jets



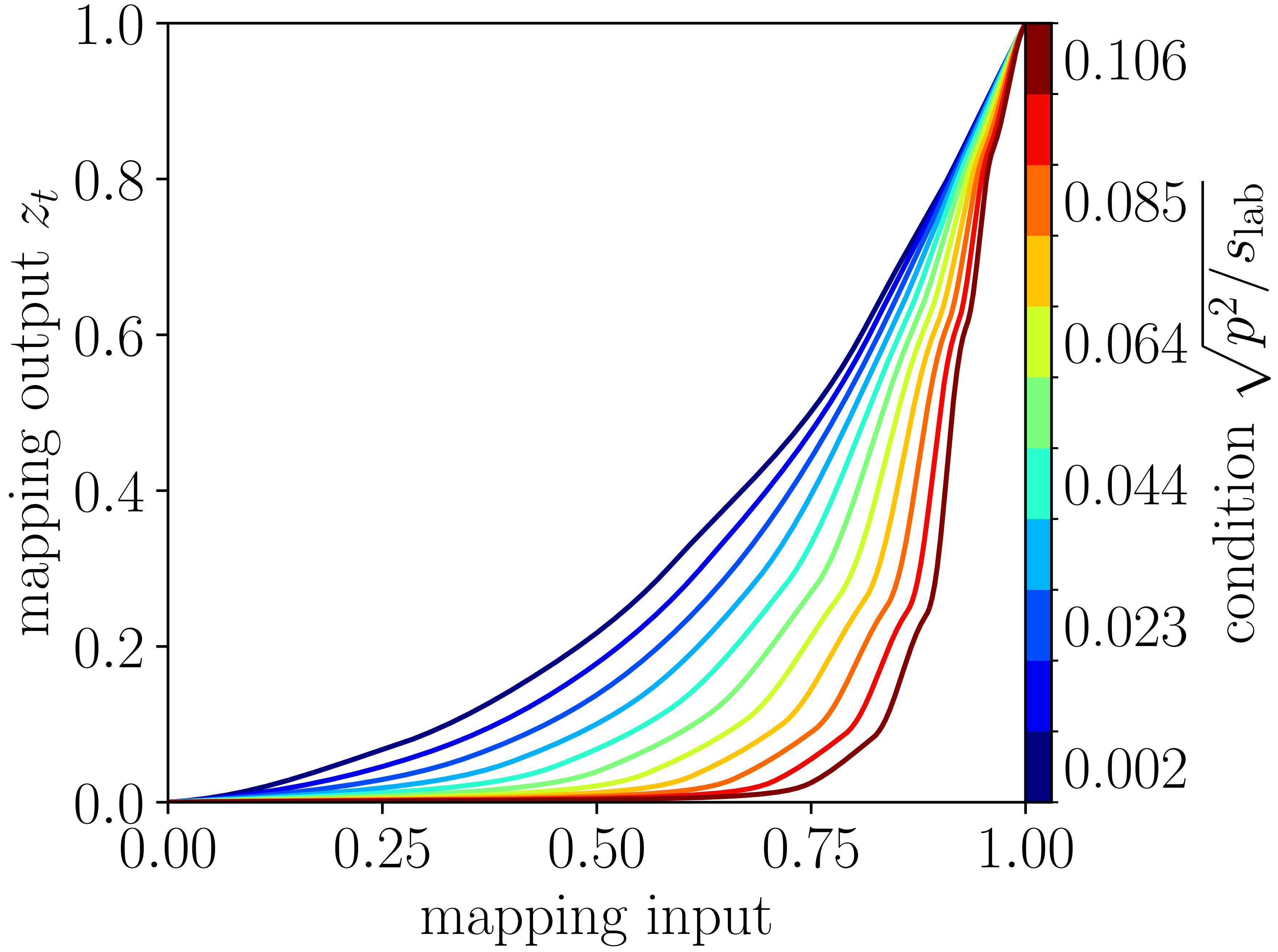
- good performance even though no channel-specific training ①
- **trained for n jets, used for n+1 jets**
  → performance like VEGAS ②
  → cheap training
- further improvements for VEGAS trained on top of MadNIS-Lite

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