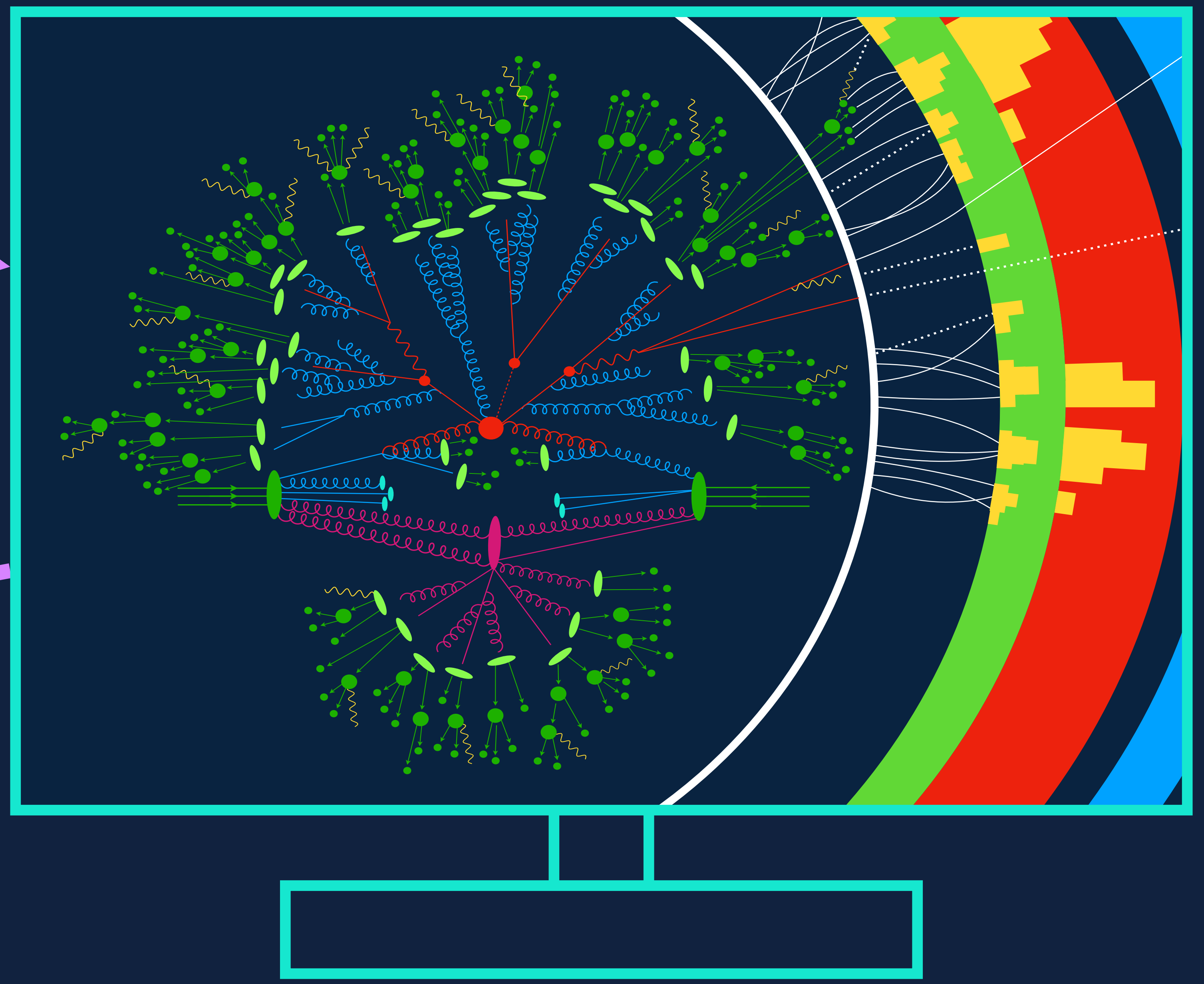
The MadNIS Reloaded

Boosting MadGraph with Neural Importance Sampling

$$
\begin{aligned}
\mathscr{L} = &-\frac{1}{4}F^{\mu\nu}F_{\mu\nu} \\
&+ i\bar{\psi}\gamma^{\mu}D_{\mu}\psi \\
&+ \bar{\psi}^{i}_{L}y_{ij}\Phi\psi^{j}_{R} + \text{h.c.} \\
&+ |D_{\mu}\Phi|^{2} + V(\Phi) + \text{BSM}
\end{aligned}
$$

MadNIS

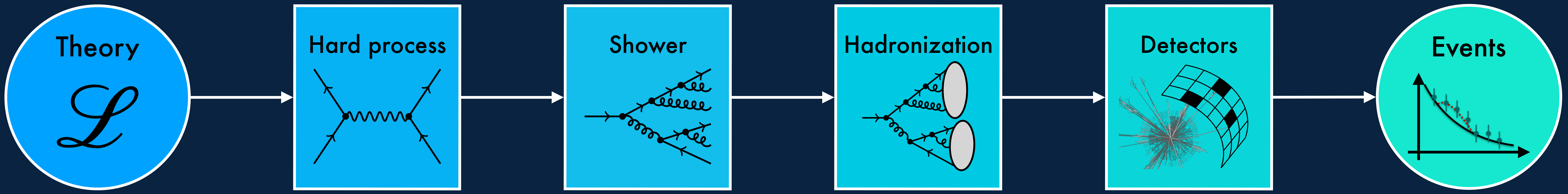


UCLouvain

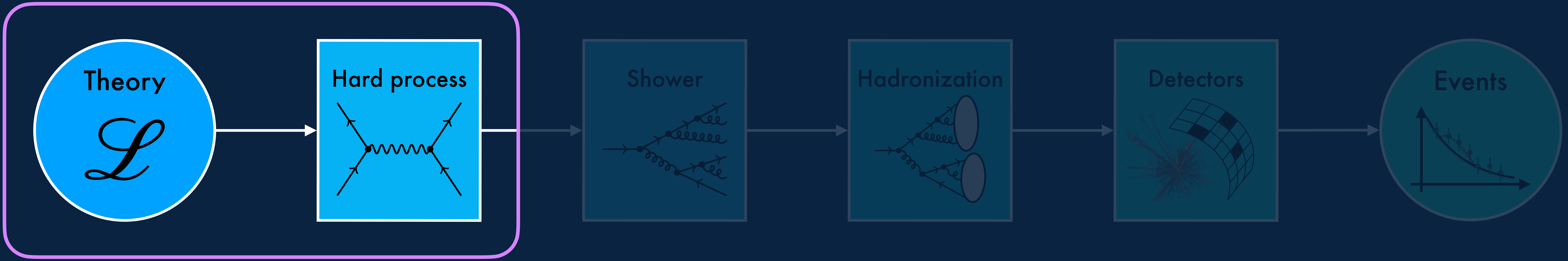
EuCAIFCon 2024 — Amsterdam

Ramon Winterhalder — UCLouvain

The LHC simulation chain



The LHC simulation chain



Differential cross section known from **QFT:**

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

Total cross section:

$$\sigma = \int_\Phi \mathrm{d}\sigma$$

Monte Carlo **integration + sampling** from differential cross section

↓

Accelerate with **Deep Generative Models**

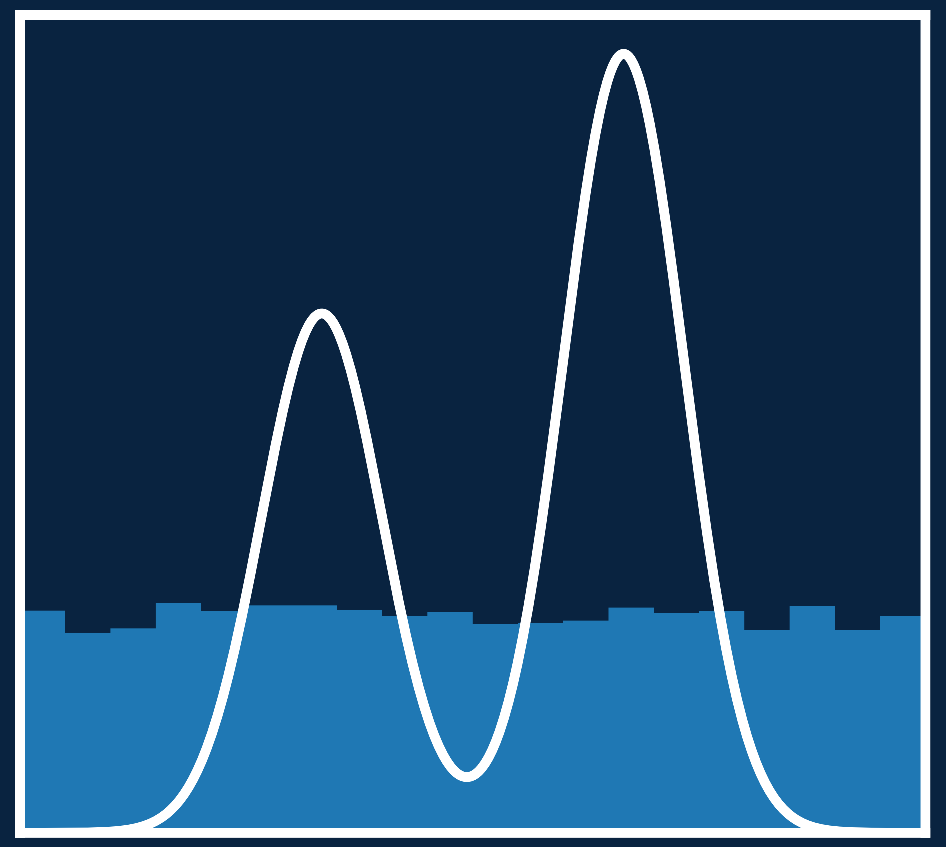
Exact sampling ensured by **known likelihood**

↓

better model
=
faster sampling

Monte Carlo Integration

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

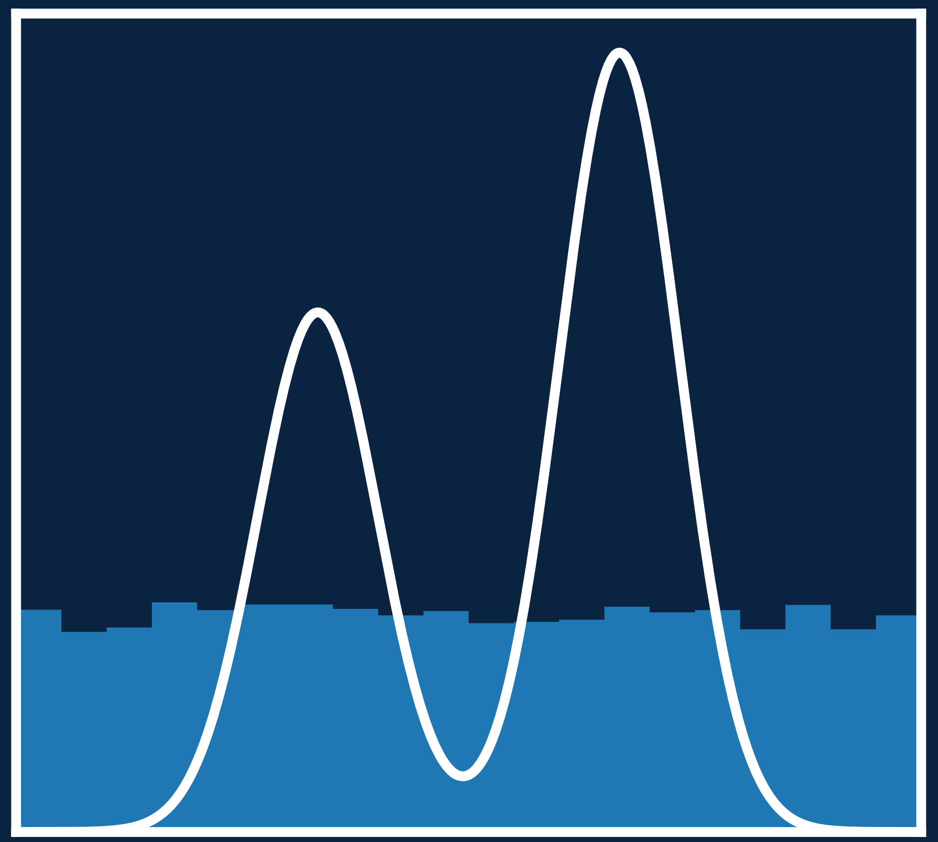


Flat sampling:
inefficient

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

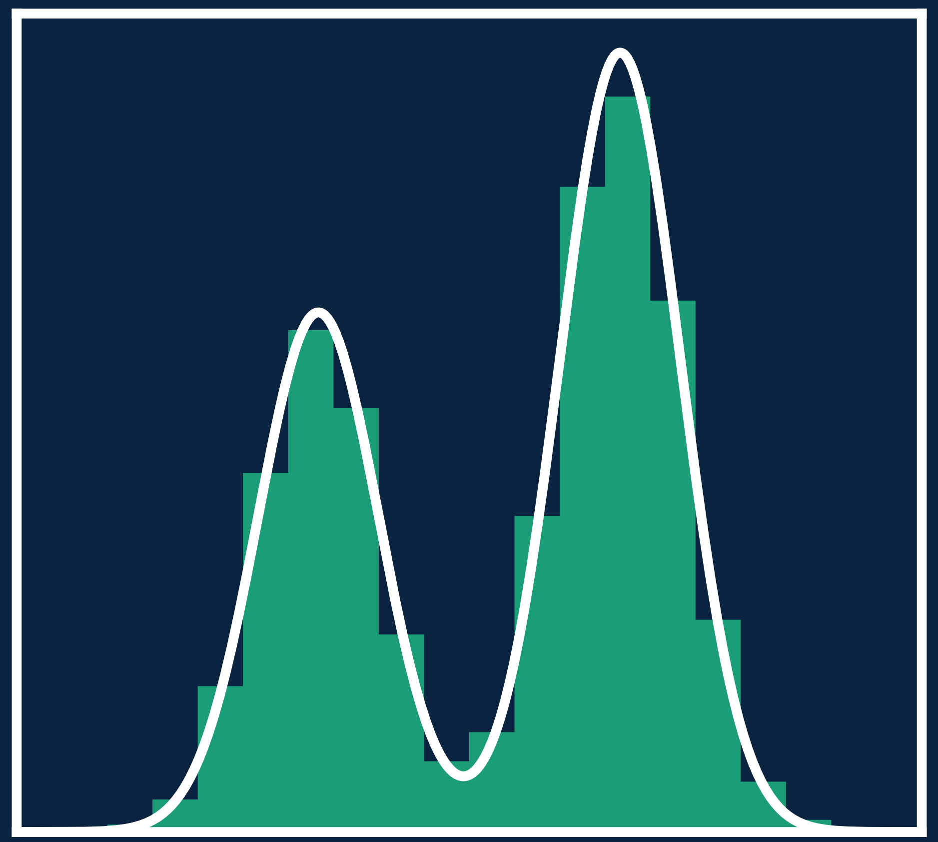
Monte Carlo Integration

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



Flat sampling:
inefficient

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

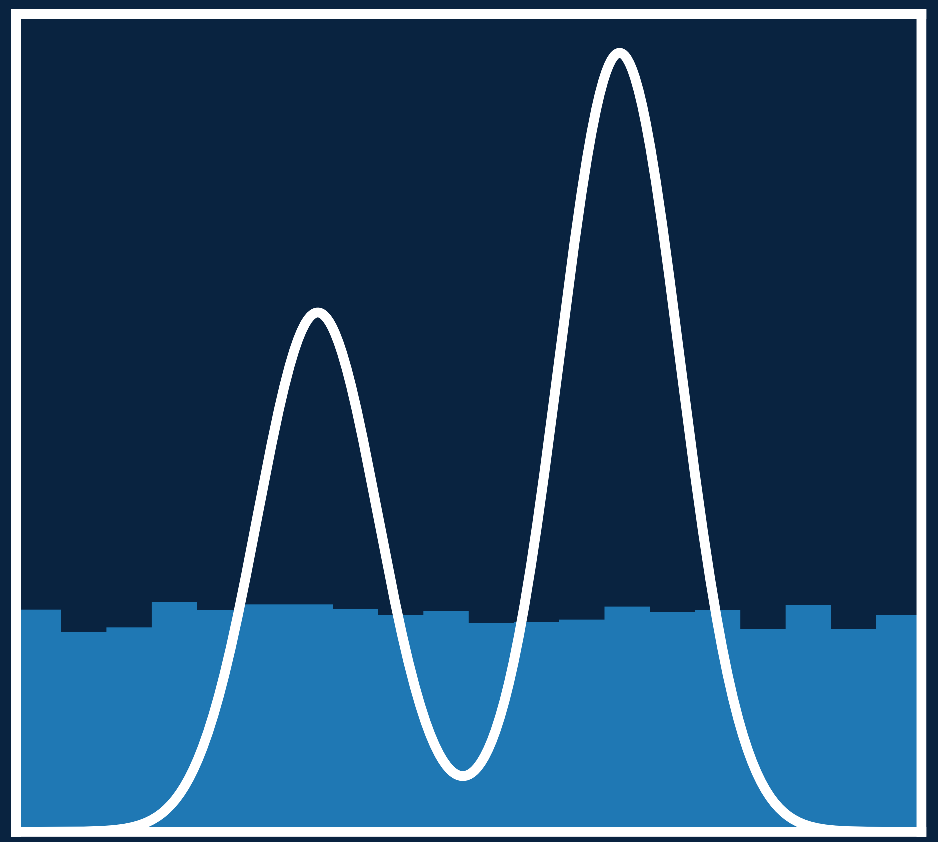


Importance sampling:
find g close to f

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

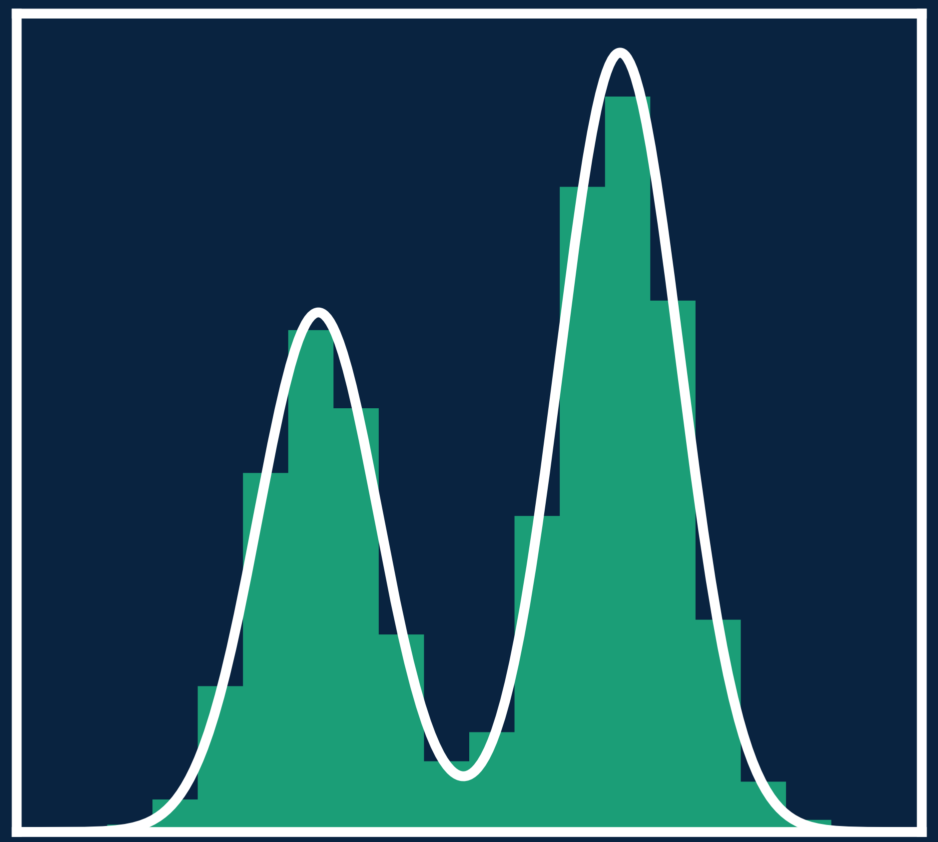
Monte Carlo Integration

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



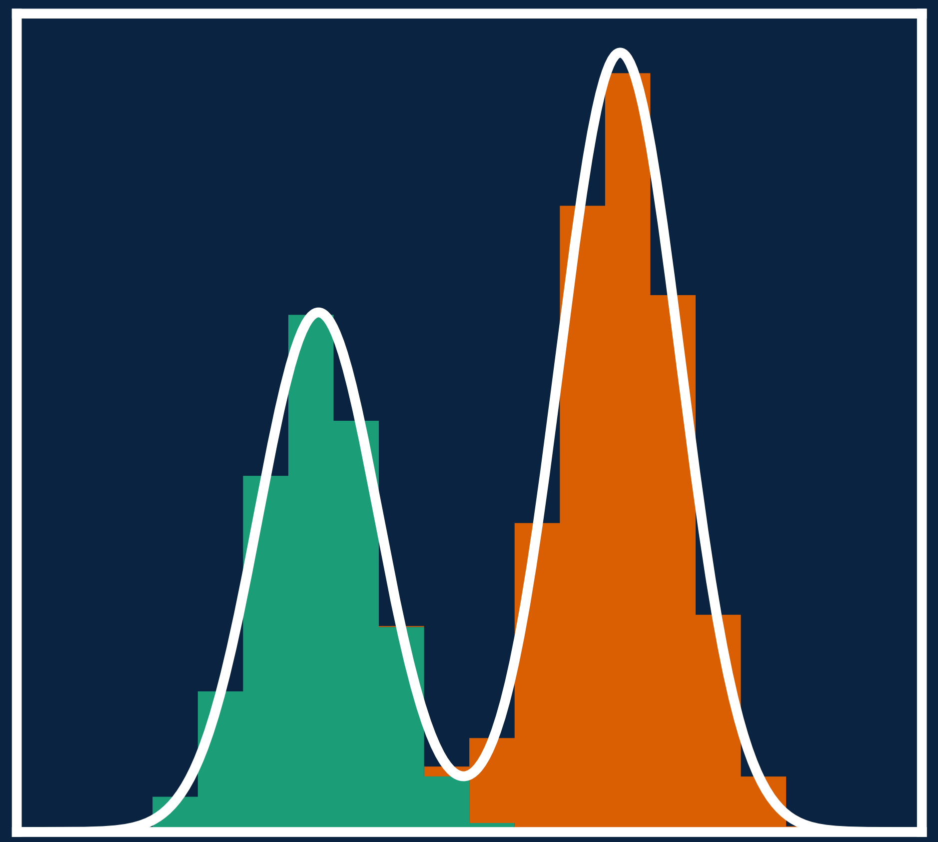
Flat sampling:
inefficient

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



Importance sampling:
find g close to f

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



Multi-channel:
one map for each channel

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

Event generation

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

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

Channel weights

MadGraph: $\alpha_i \sim |M_i|^2$

or

$$\alpha_i \sim \prod_k |p_k^2 - m_k^2 - \mathrm{i} m_k \Gamma_k|^{-2}$$

Channel mappings

MadGraph: use amplitude structure, …

refine with **VEGAS**

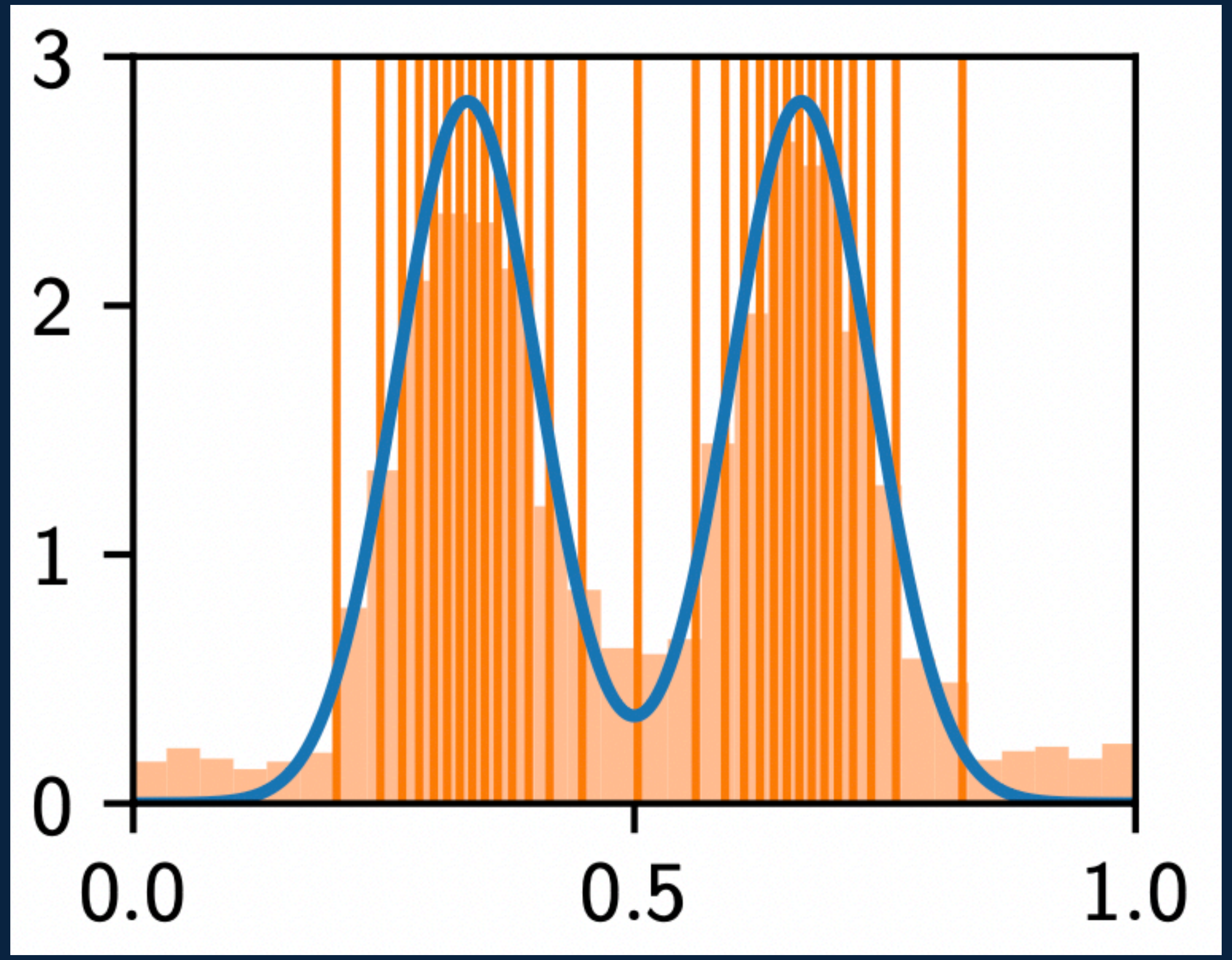
(factorized, histogram based importance sampling)

Importance sampling — VEGAS

Factorize probability

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

Fit bins with equal probability
and varying width



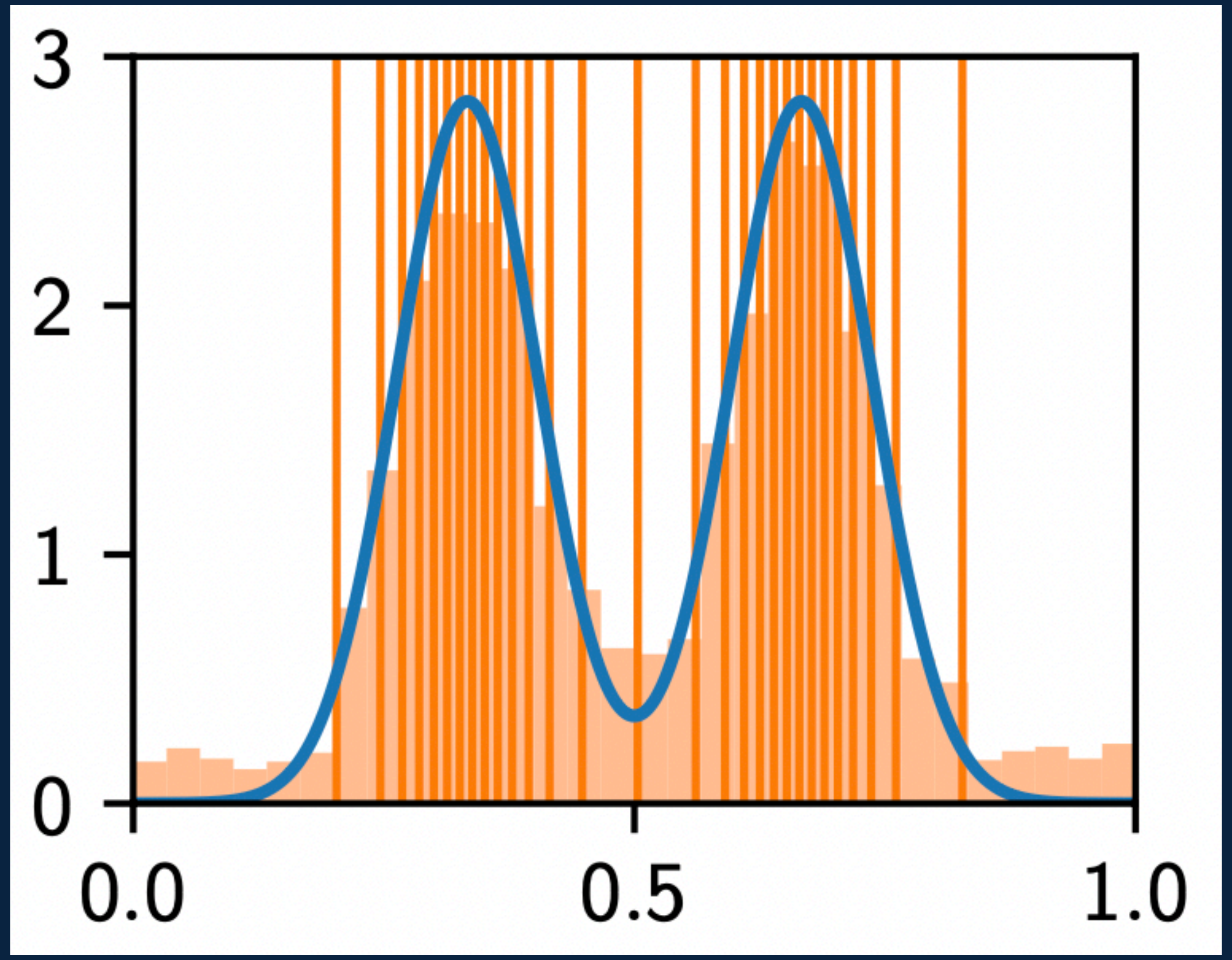
[G. P. Lepage, 1978]

Importance sampling — VEGAS

Factorize probability

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

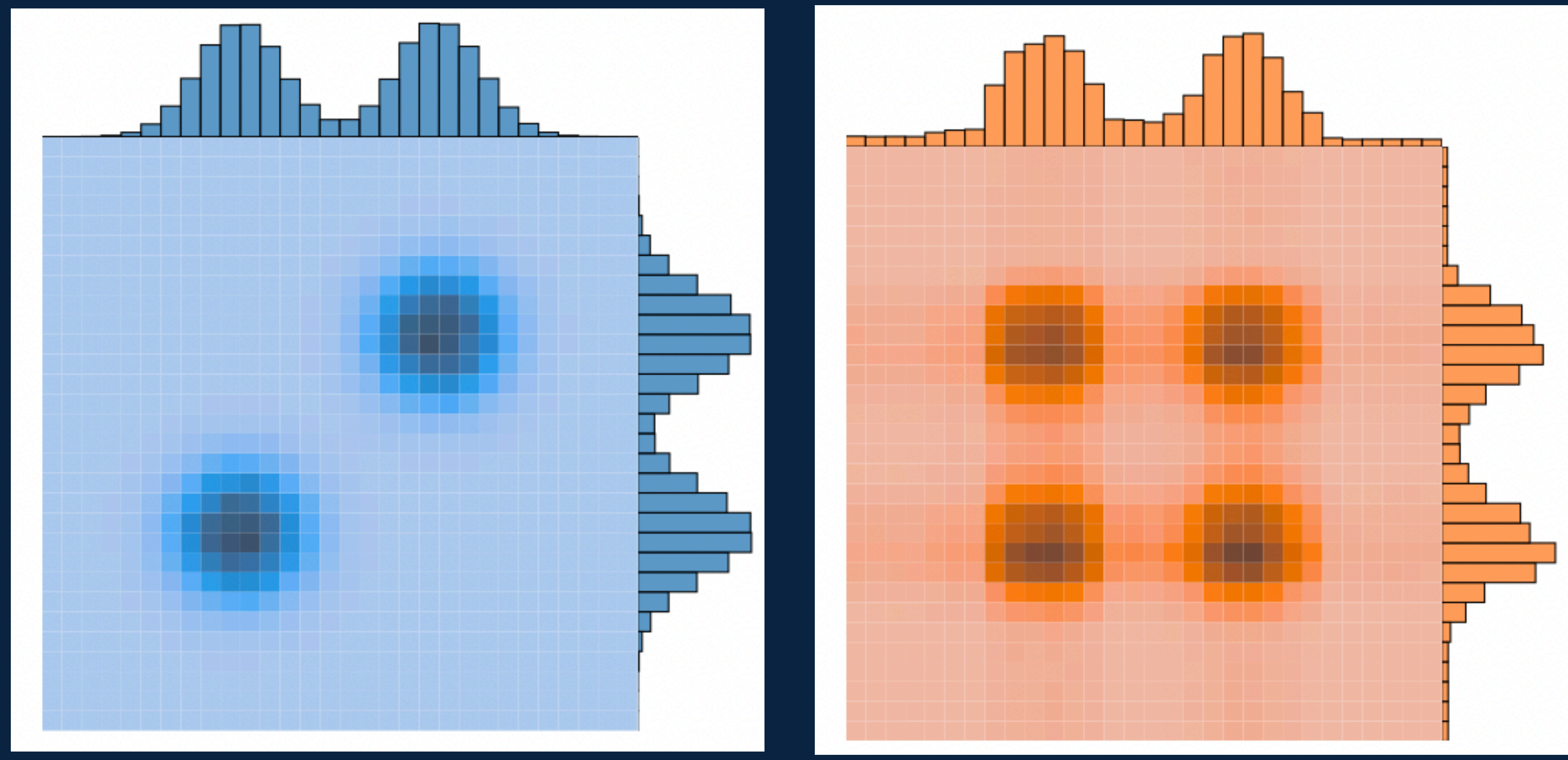
Fit bins with equal probability
and varying width



⊕ Computationally cheap

⊖ High-dim and rich peaking functions
→ **slow convergence**

⊖ Peaks not aligned with grid axes
→ **phantom peaks**



[G. P. Lepage, 1978]

MadNIS

Neural Importance Sampling

Heimel, Huetsch, Maltoni, Mattelaer, Plehn, RW [2311.01548]
Heimel, RW, Butter, Isaacson, Krause, Maltoni, Mattelaer, Plehn [2212.06172]

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 channel and mappings

Normalizing flow to refine channel mappings

Fully connected network to refine channel weights

Update simultanously with variance as loss function

Overview

MadNIS

Basic functionality

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

Improved multi-channeling

- Stratified sampling/ training
- Symmetries between channels
- Channel Dropping
- Partial weight buffering

Improved training

- VEGAS Initialization
- Buffered Training
- Surrogate Integrand

Overview

MadNIS

Basic functionality

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

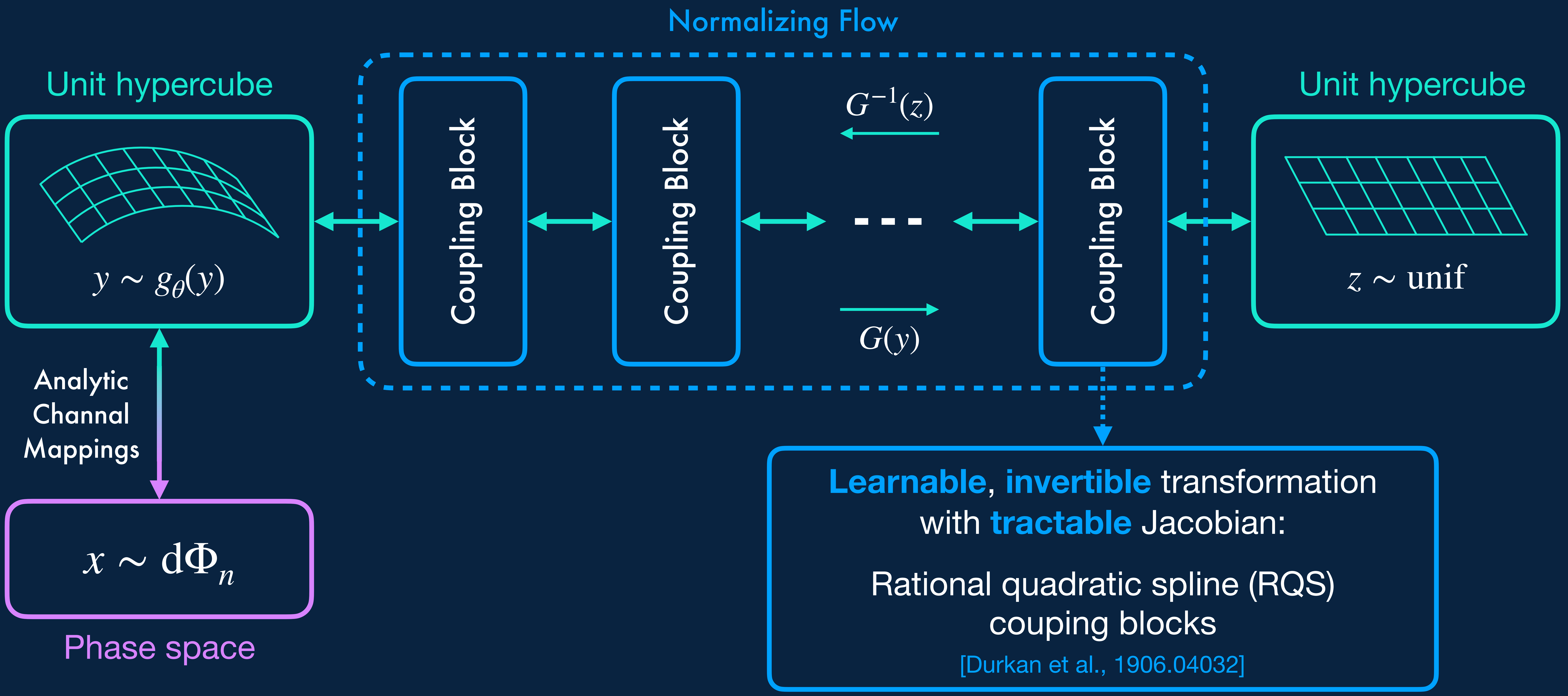
Improved multi-channeling

- Stratified sampling/ training
- Symmetries between channels
- Channel Dropping
- Partial weight buffering

Improved training

- VEGAS Initialization
- Buffered Training
- Surrogate Integrand

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]

Overview

Basic functionality

Neural Channel Weights

Normalizing Flow

MadGraph matrix elements

MadEvent channel mappings

MadNIS

Improved multi-channeling

Stratified sampling/ training

Symmetries between channels

Channel Dropping

Partial weight buffering

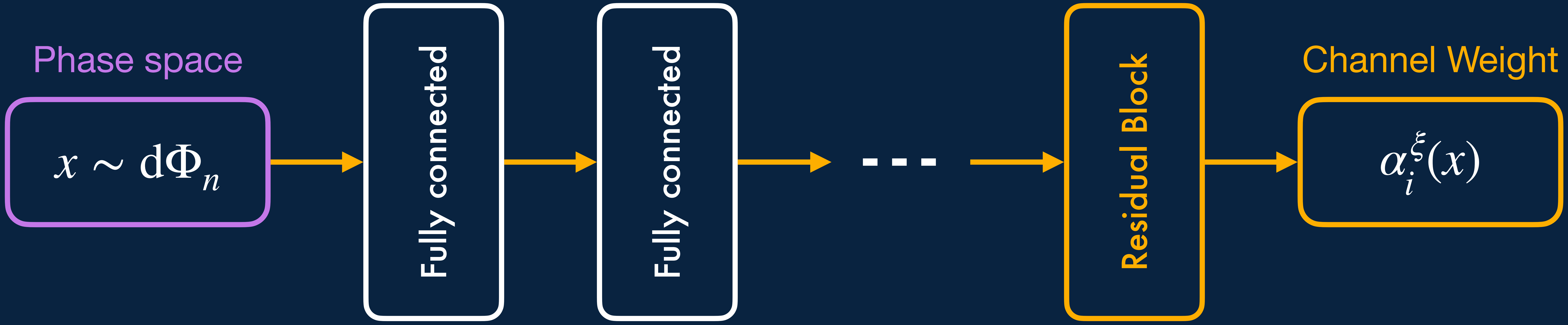
Improved training

VEGAS Initialization

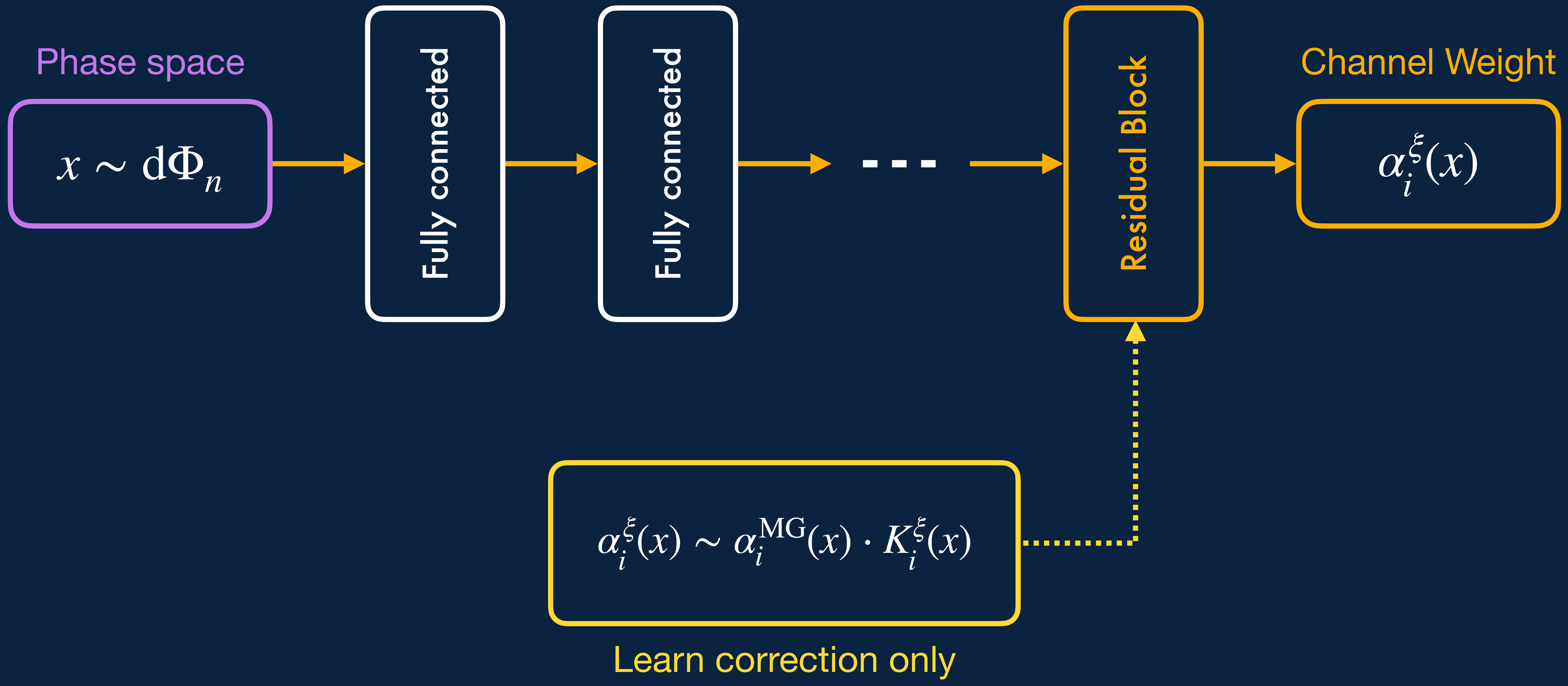
Buffered Training

Surrogate Integrand

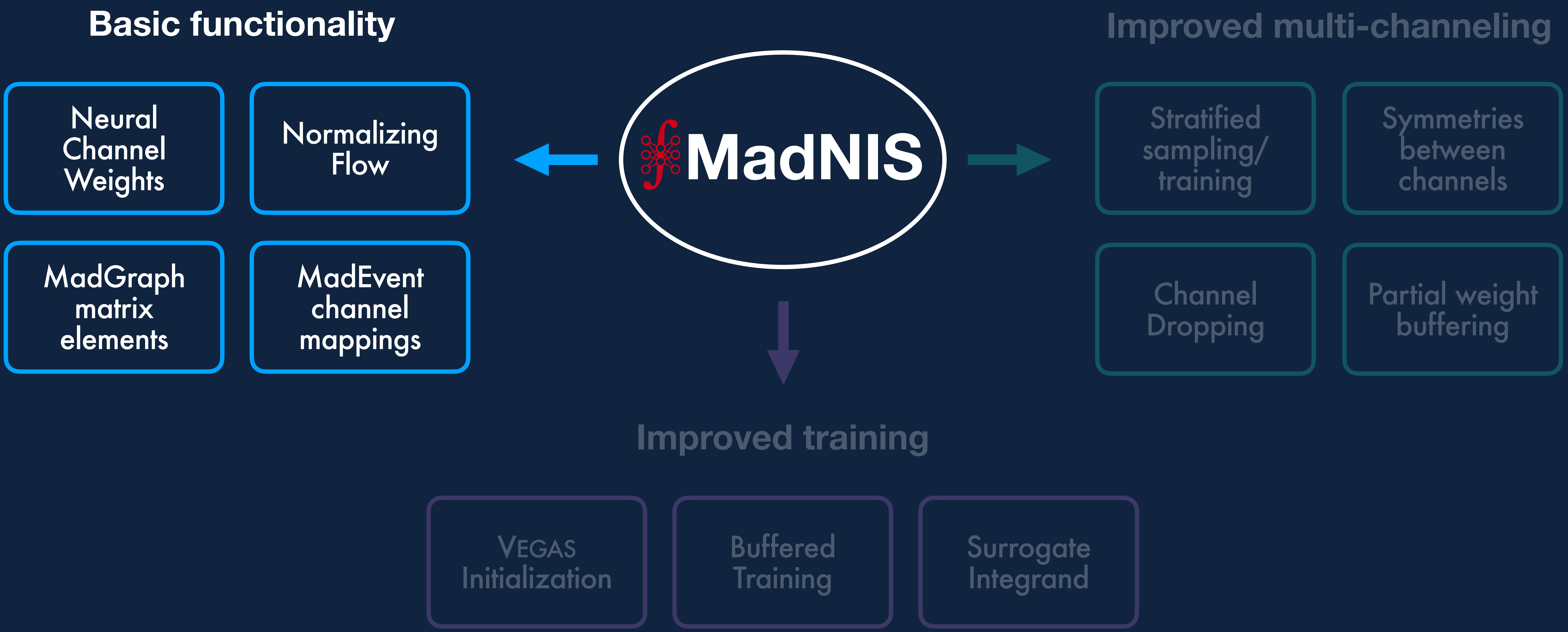
Neural Channel Weights



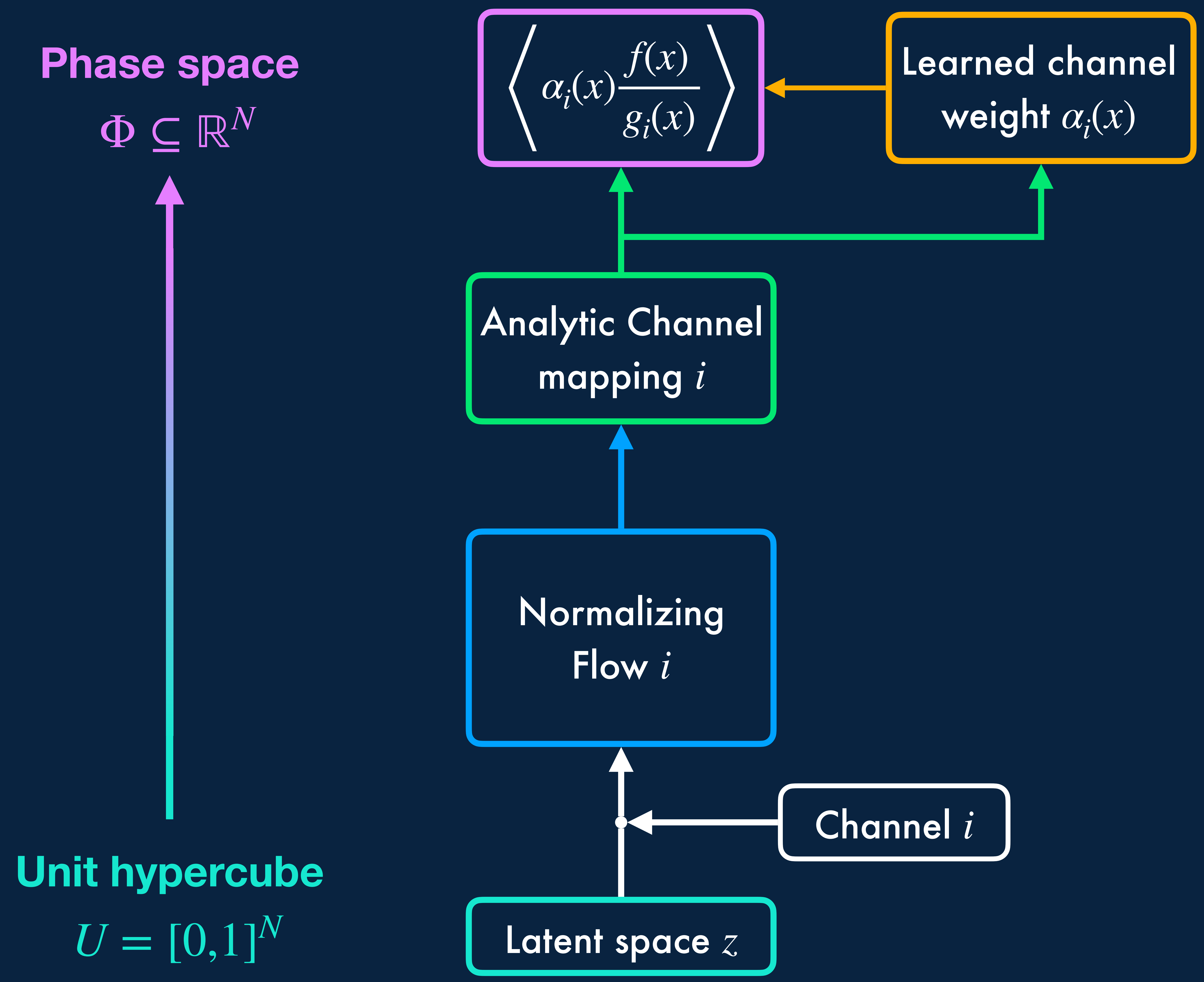
Neural Channel Weights



Overview


Basic functionality
Neural Channel Weights
Normalizing Flow
MadGraph matrix elements
MadEvent channel mappings
MadNIS
Improved multi-channeling
Stratified sampling/ training
Symmetries between channels
Channel Dropping
Partial weight buffering
Improved training
VEGAS Initialization
Buffered Training
Surrogate Integrand

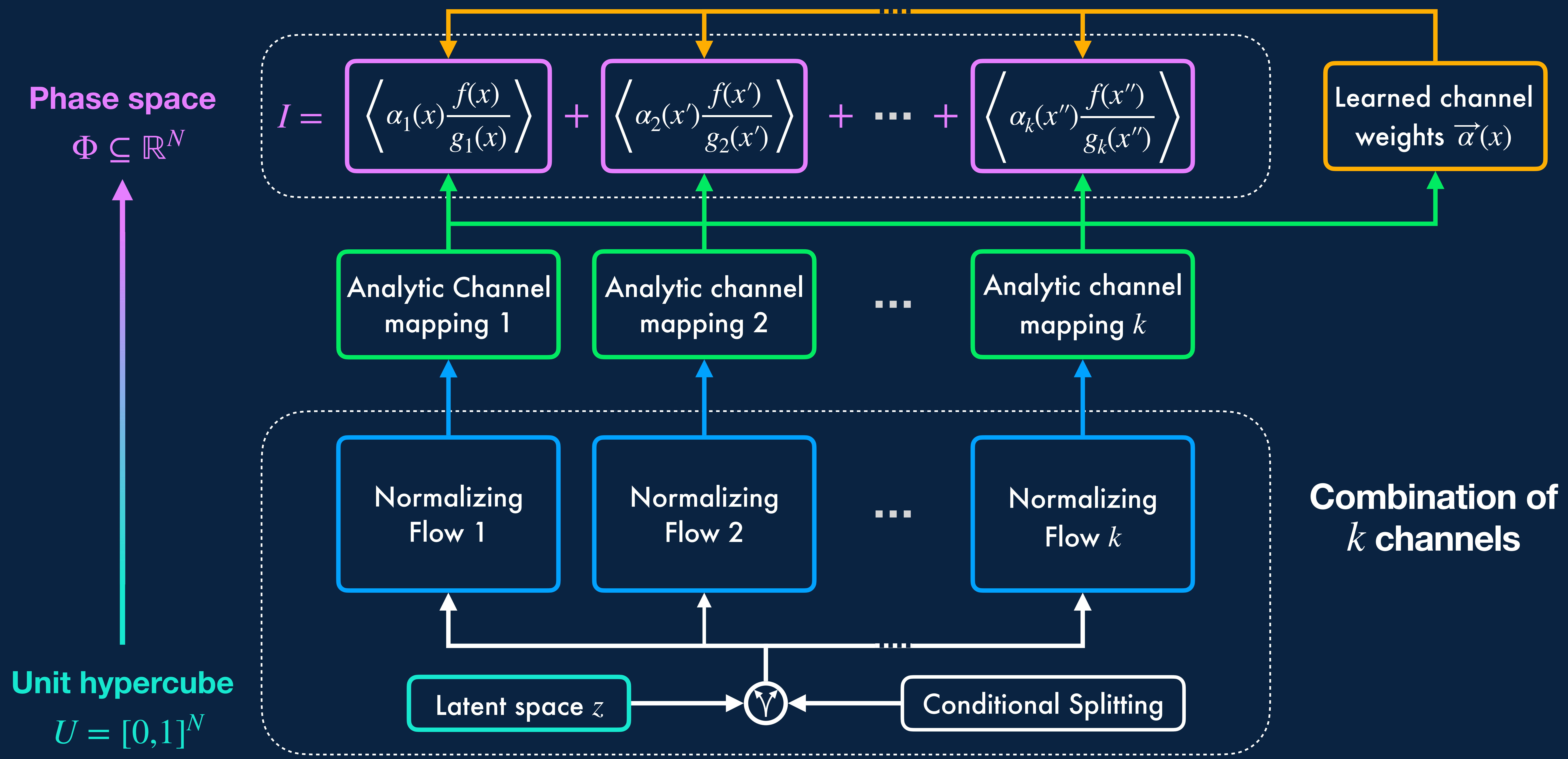
MadNIS — Neural Importance Sampling



Single channel i

MadNIS — Neural Importance Sampling

15



Loss function

Minimize total variance

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

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

Total variance depends on N_i

↓

affects optimal $\alpha_i(x)$

↓

use stratified sampling

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

Loss function

Minimize total variance

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

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

Total variance depends on N_i

$\downarrow$

affects optimal $\alpha_i(x)$

$\downarrow$

use stratified sampling

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

MadNIS loss function

$$\mathscr{L} = \sigma_{\text{tot}}^2 = \left(\sum_i \sigma_i \right)^2$$

Overview

MadNIS

Basic functionality

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

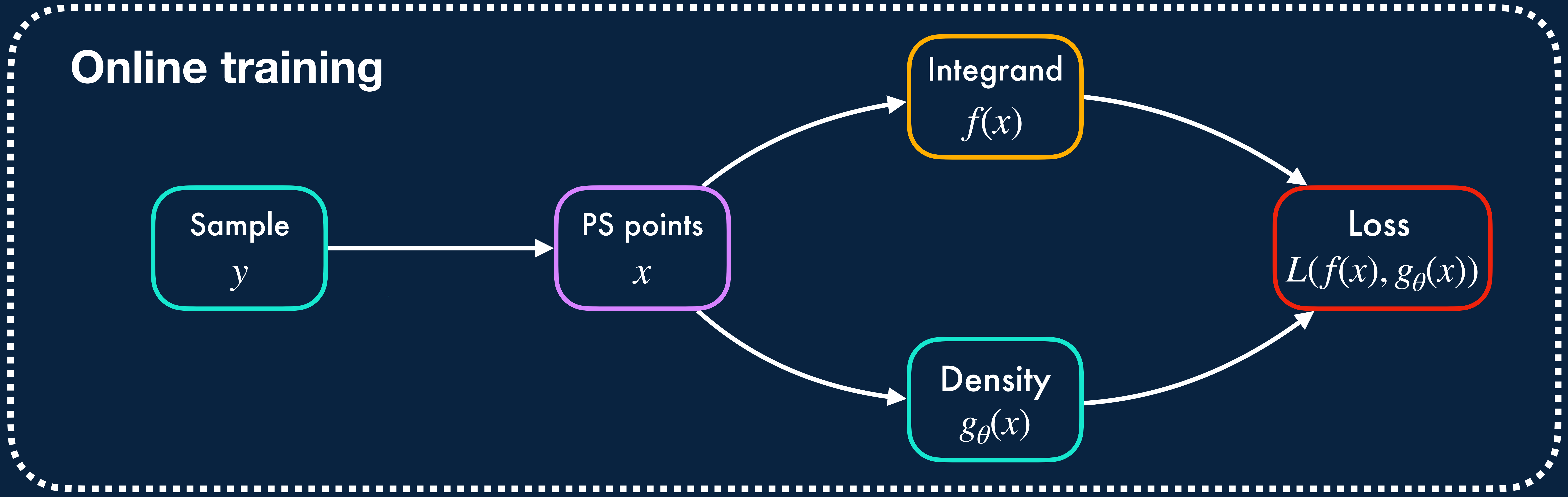
Improved multi-channeling

- Stratified sampling/ training
- Symmetries between channels
- Channel Dropping
- Partial weight buffering

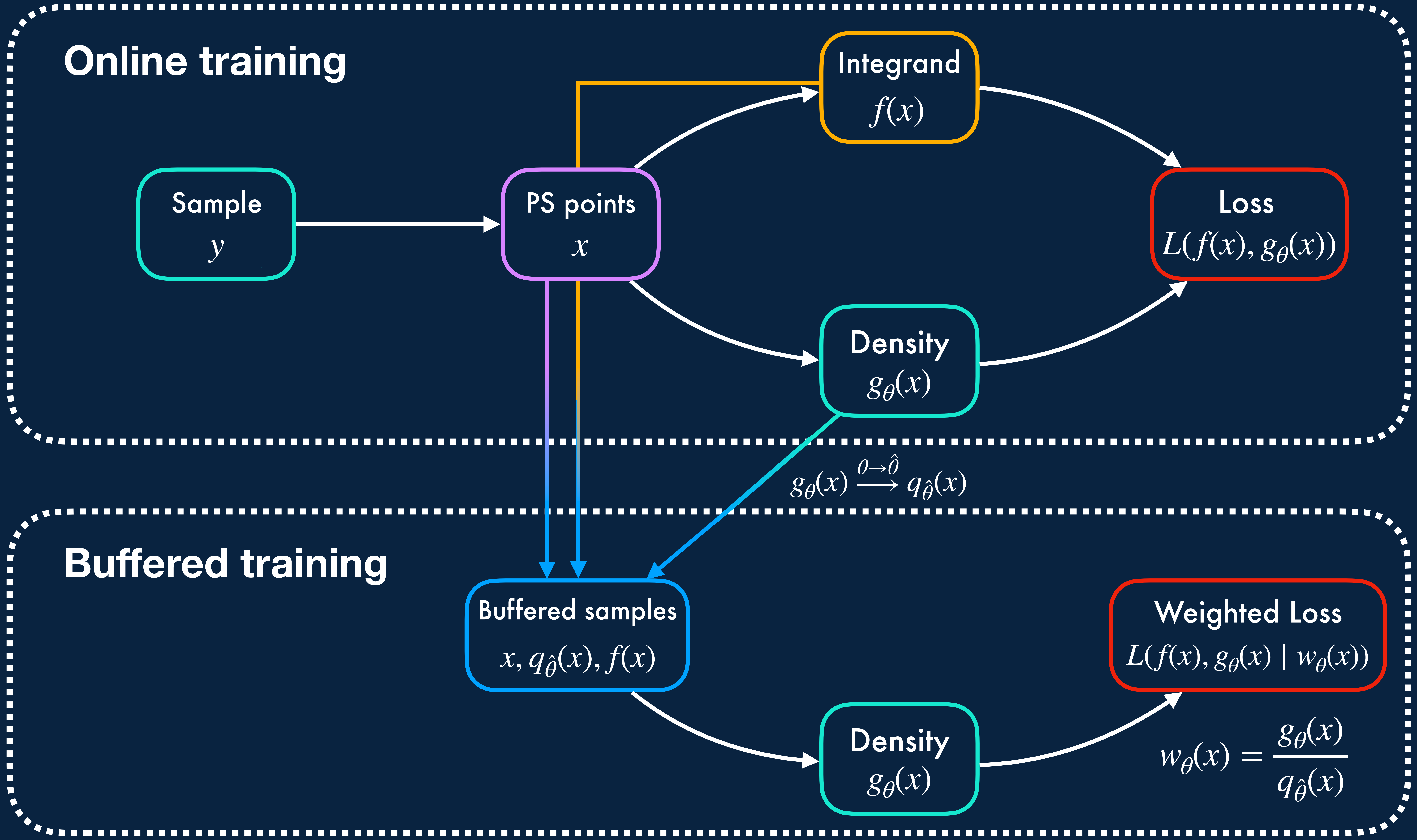
Improved training

- VEGAS Initialization
- Buffered Training
- Surrogate Integrand

Buffered training



Buffered training



Buffered training

Training algorithm

generate new samples, train on them, save samples

↓

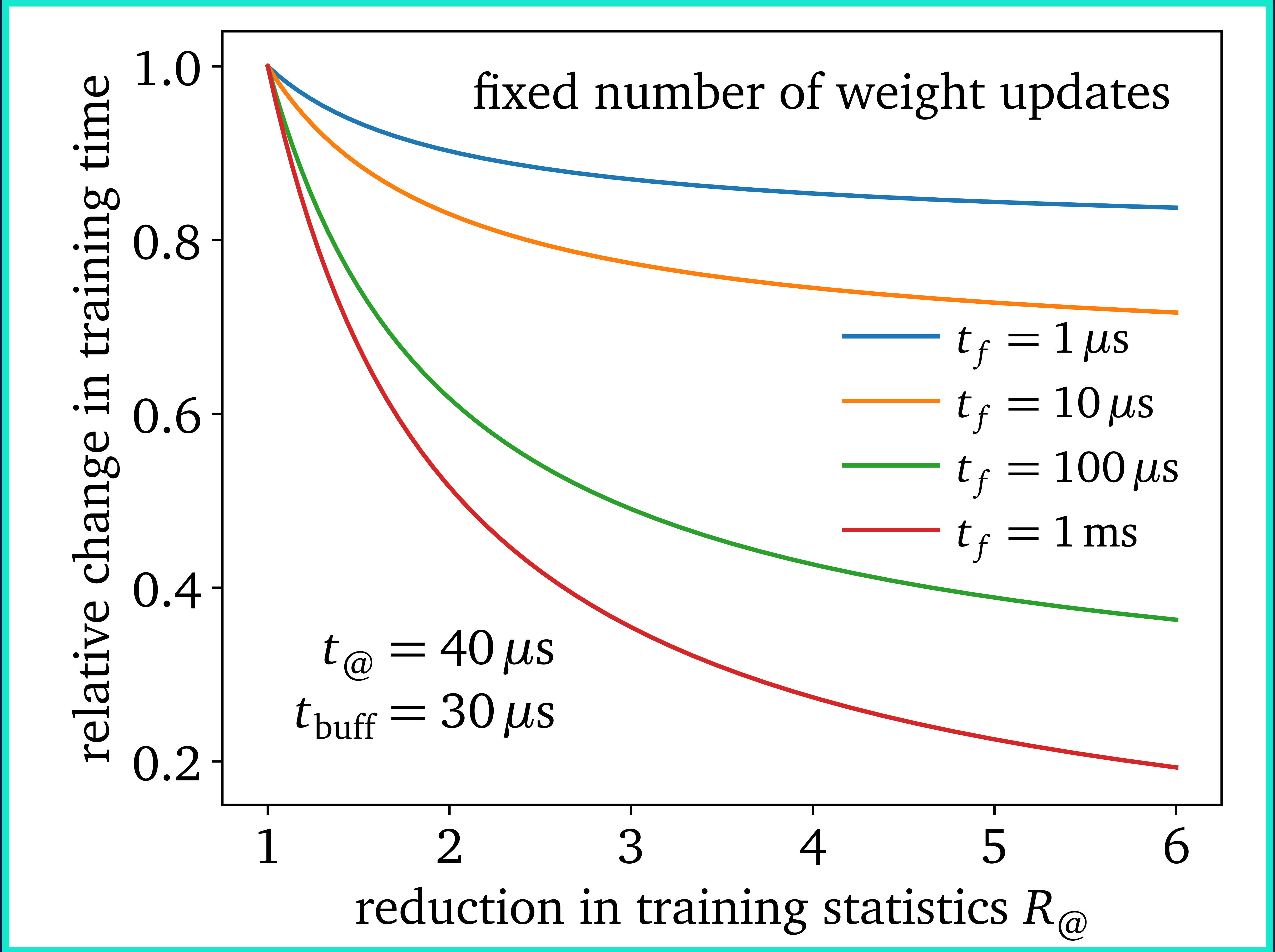
train on saved samples n times

↓

repeat

Reduction in training statistics by

$$R_@ = n + 1$$



Overview

Basic functionality

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

MadNIS

Improved multi-channeling

- Stratified sampling/training
- Symmetries between channels
- Channel Dropping
- Partial weight buffering

Improved training

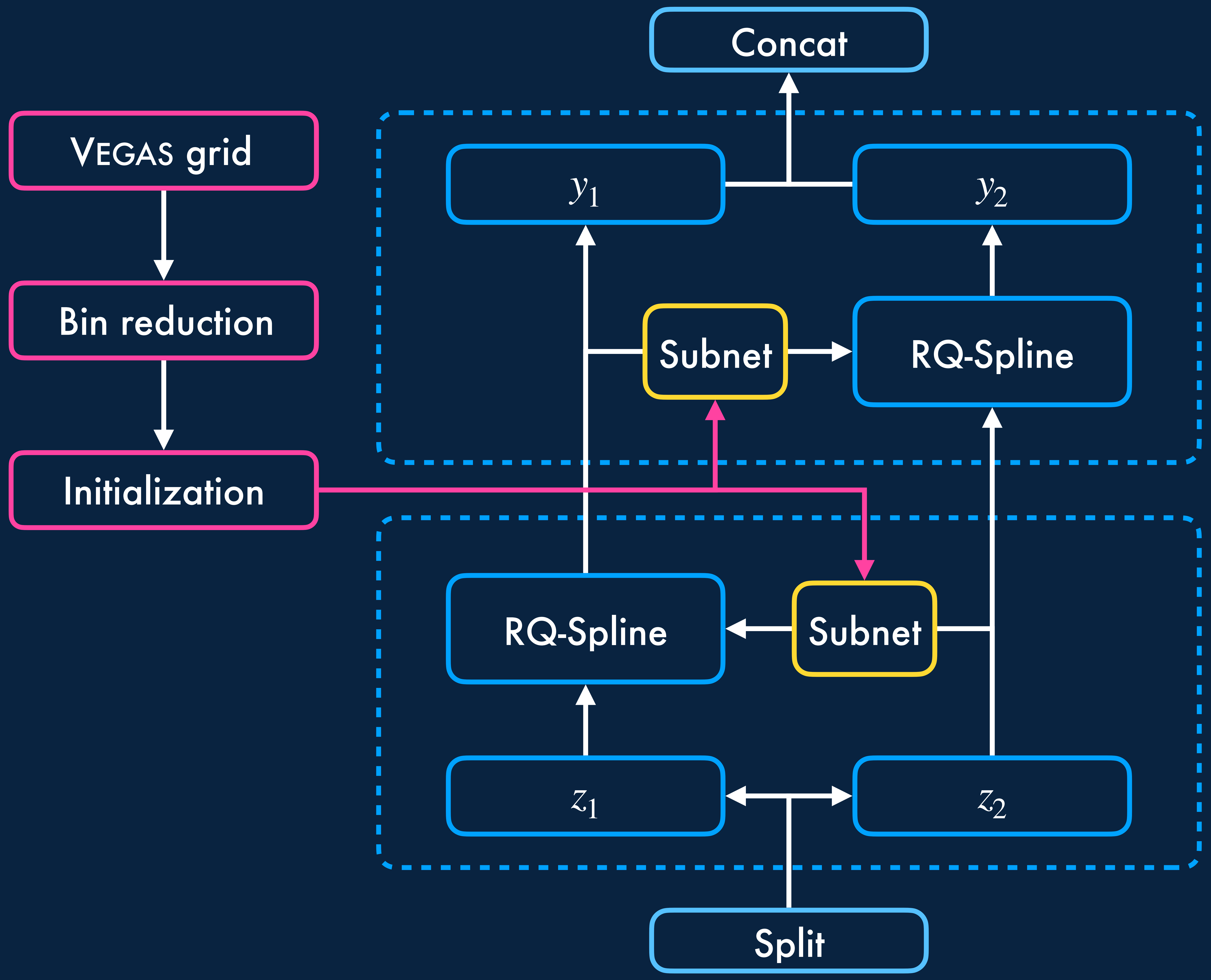
- VEGAS Initialization
- Buffered Training
- Surrogate Integrand

VEGAS initialization

	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

Combine advantages:

Pre-trained VEGAS grid as starting point for flow training



Overview

MadNIS

Basic functionality

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

Improved multi-channeling

- Stratified sampling/ training
- Symmetries between channels
- Channel Dropping
- Partial weight buffering

Improved training

- VEGAS Initialization
- Buffered Training
- Surrogate Integrand

Improved multi-channeling

Use symmetries

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

↓

use **common flow** 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

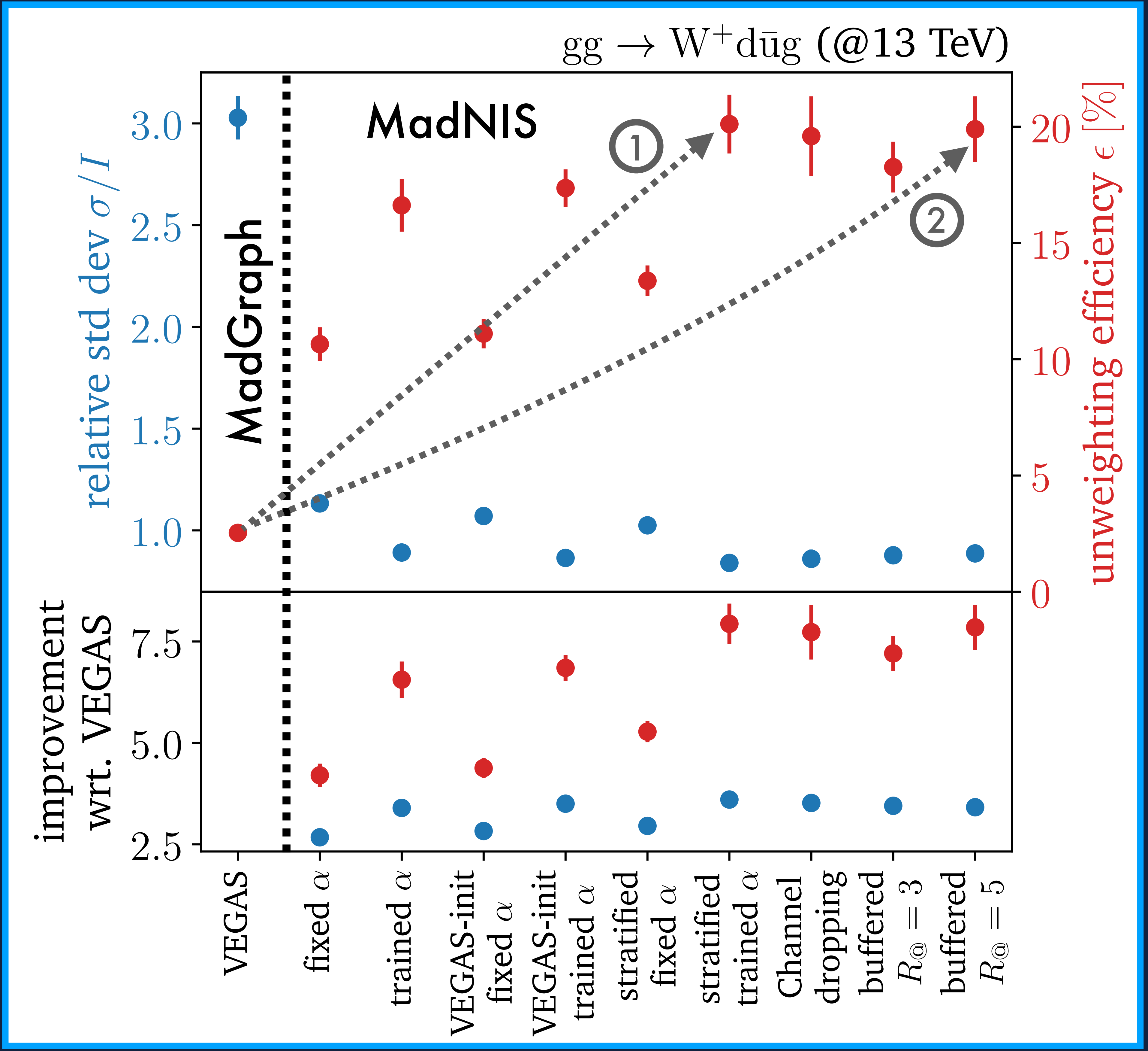
MadNIS often **reduces contribution** of some channels to total integral

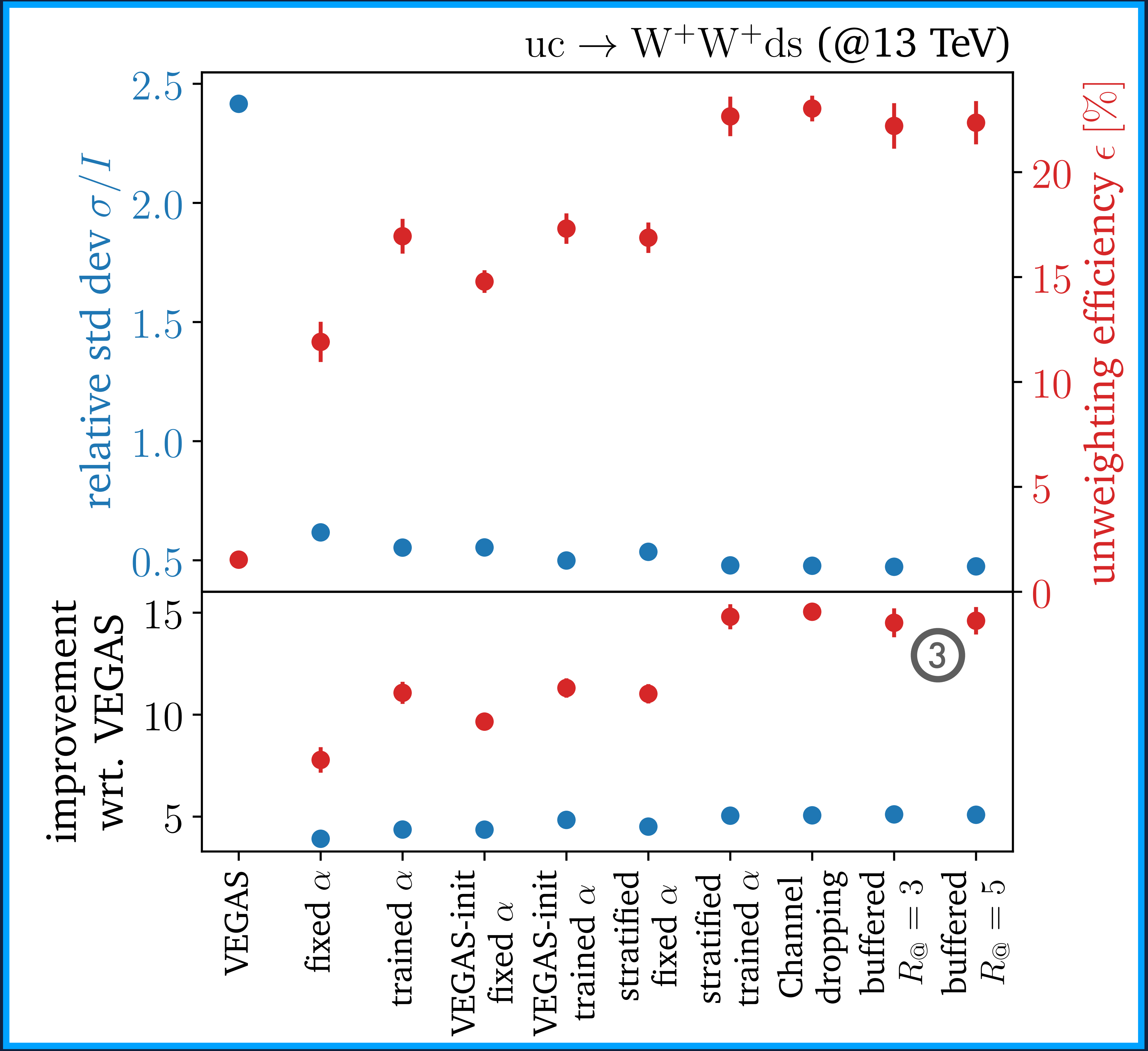
↓

remove these channels from the **training** completely

Reduced complexity
Improved stability

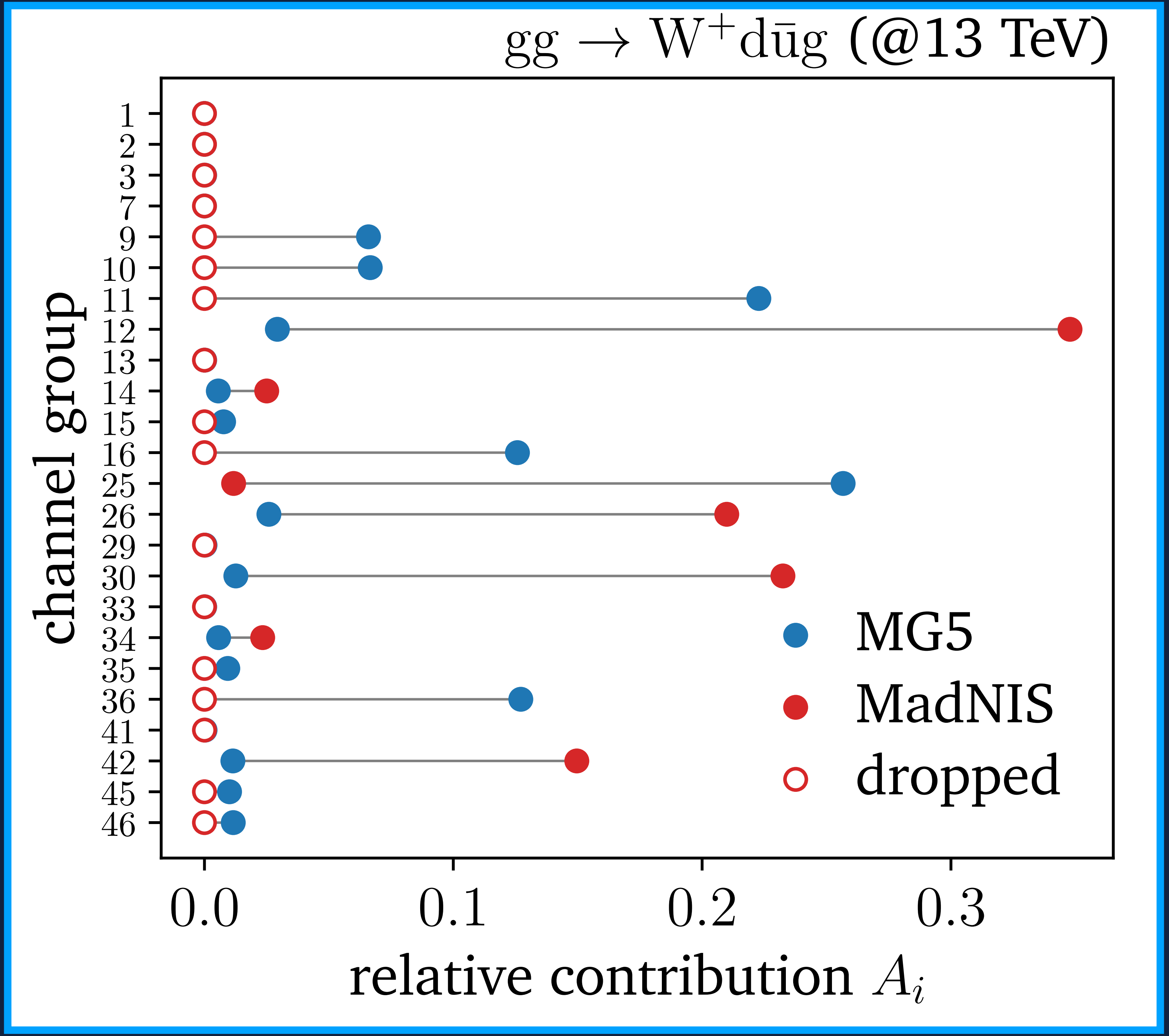
LHC processes

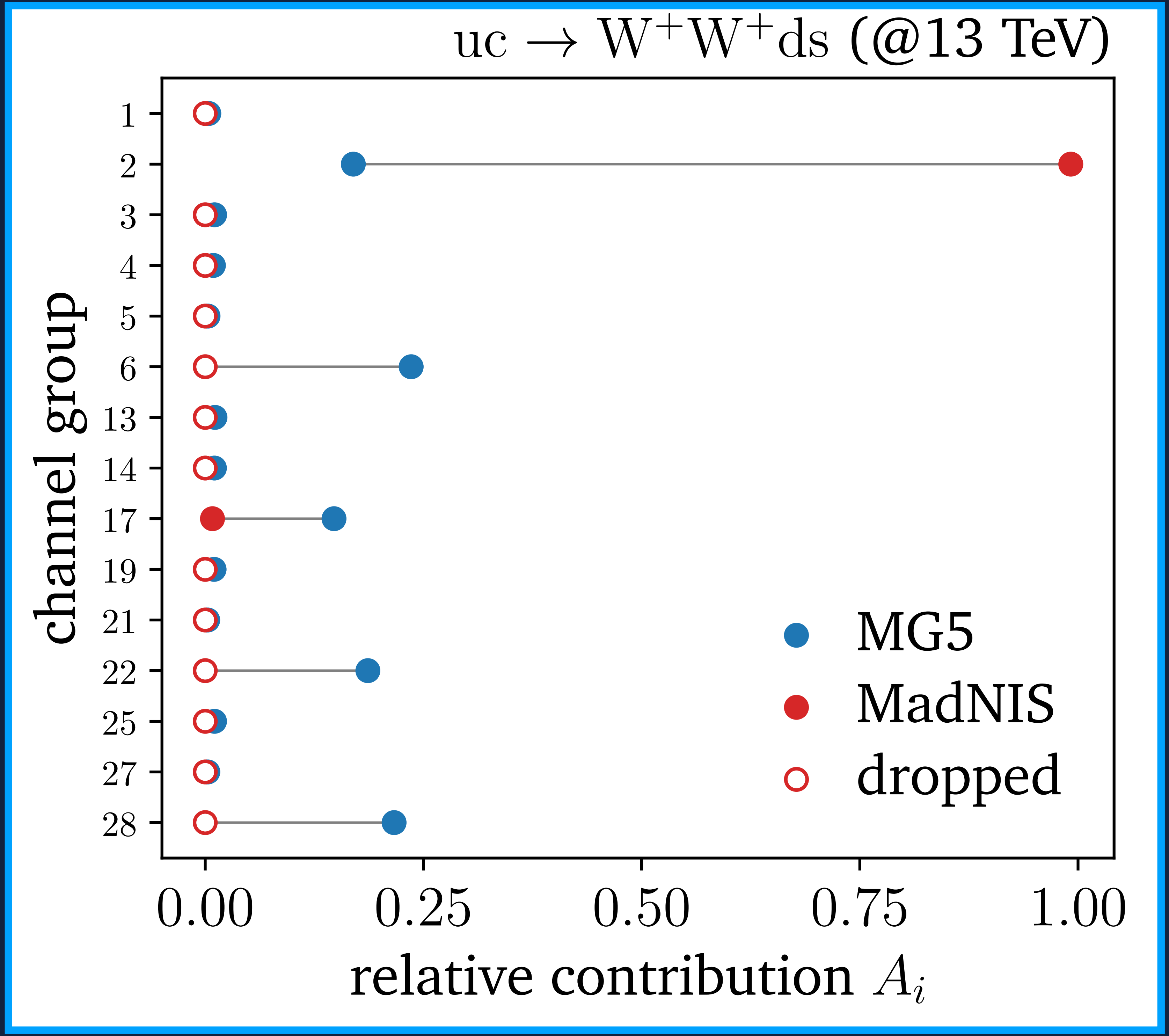




1. excellent results with all improvements
2. same performance with buffered training
3. Larger improvements for processes with large interference terms

Learned channel weights





In MadNIS many channels **are zero**

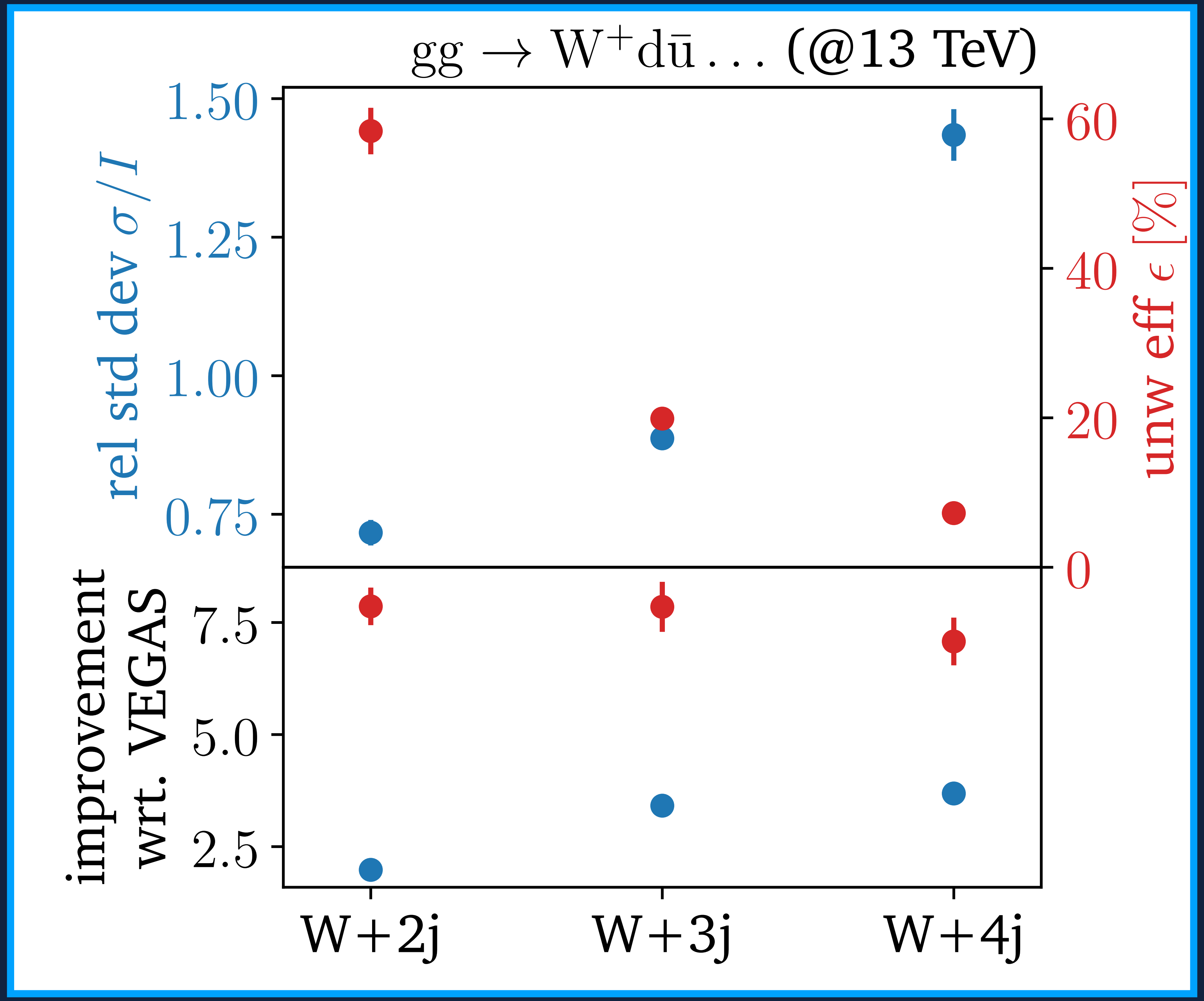
↓

droppig channels

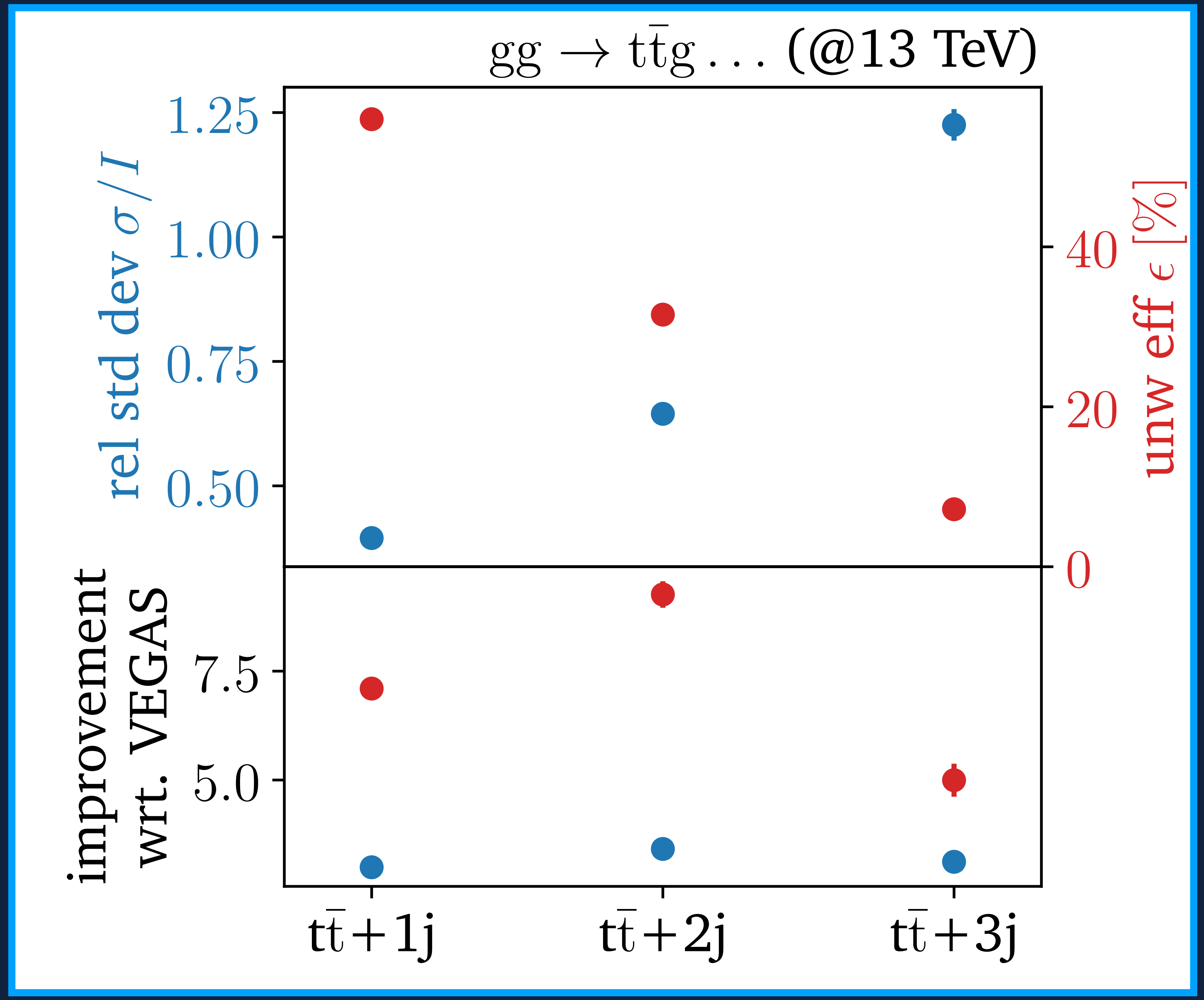
↓

more efficient training and event generation

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!

Outlook

The MadNIS Reloaded

Large improvements, even for high multiplicites and complicated processes!

[2311.01548]

Future plans

Make MadNIS part of next MadGraph version