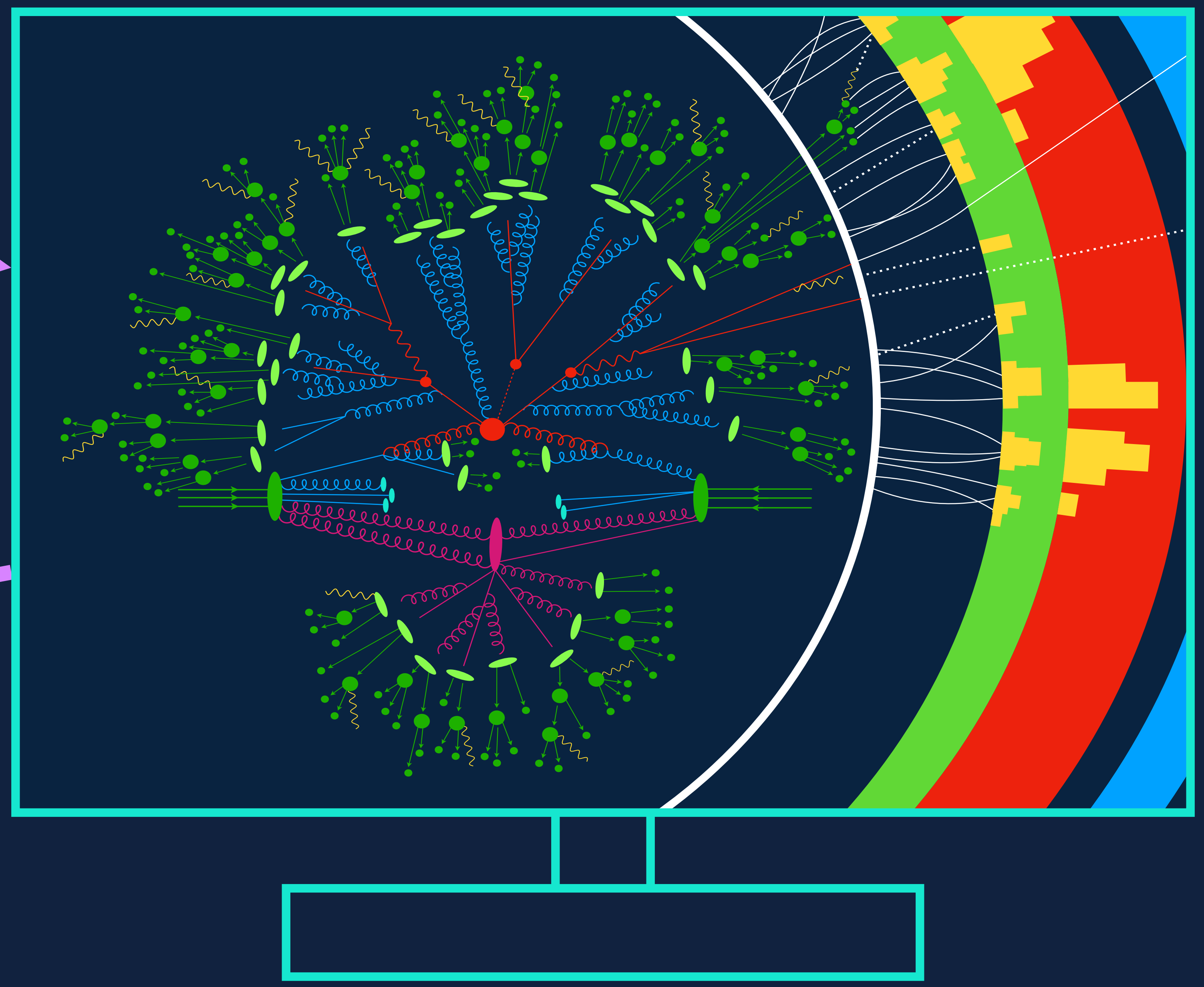
The MadNIS Reloaded

Enhancing 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

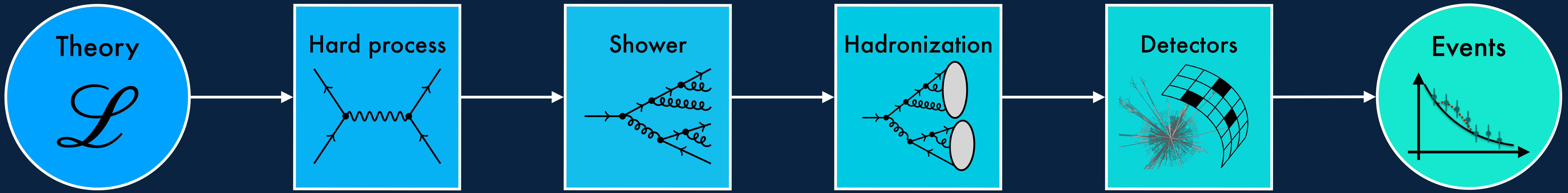


UCLouvain

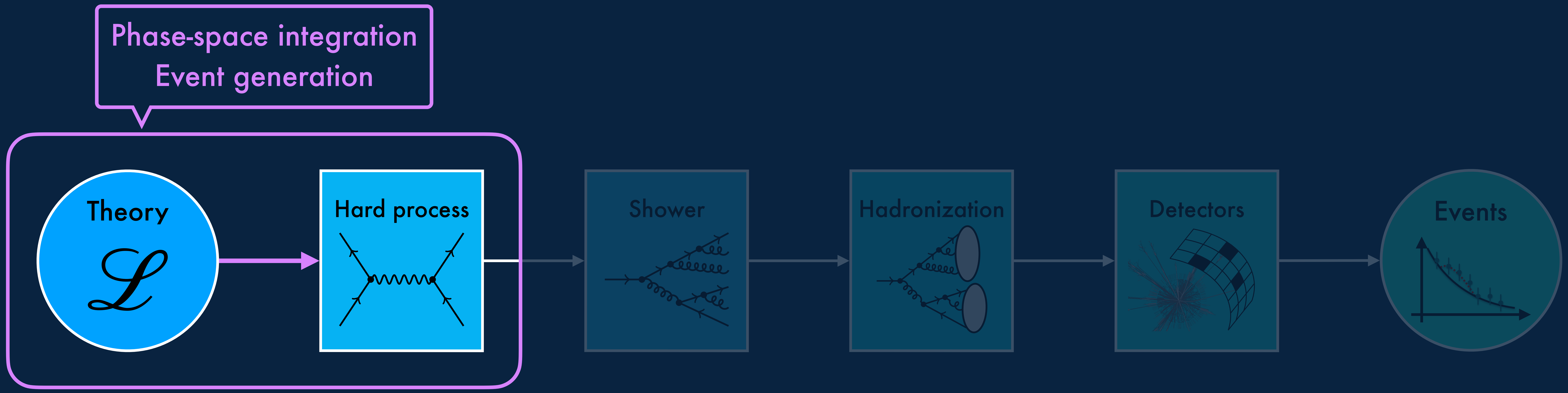
COMETA WG2 Meeting — March 2024

Ramon Winterhalder — UCLouvain

The LHC simulation chain



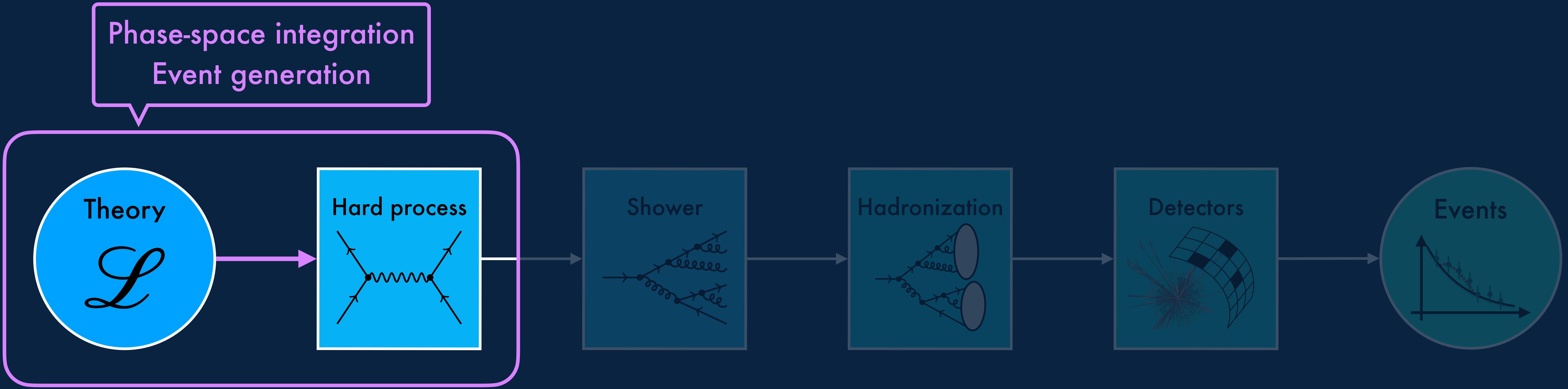
The LHC simulation chain + ML



Importance sampling
BDT [1707.00028], NN [1810.11509, 2009.07819]
NF [2001.05486, 2001.05478, 2001.10028, 2005.12719, 2112.09145, 2212.06172, 2311.01548, 2401.09069]
Chili [2302.10449]

Surrogate regression
Full weight [2109.11964],
Matrix element [1912.11055, 2002.07516, 2006.16273, 2106.09474, 2107.06625, 2109.11964, 2206.14831, 2301.13562, 2302.04005, 2306.07726]

The LHC simulation chain + ML



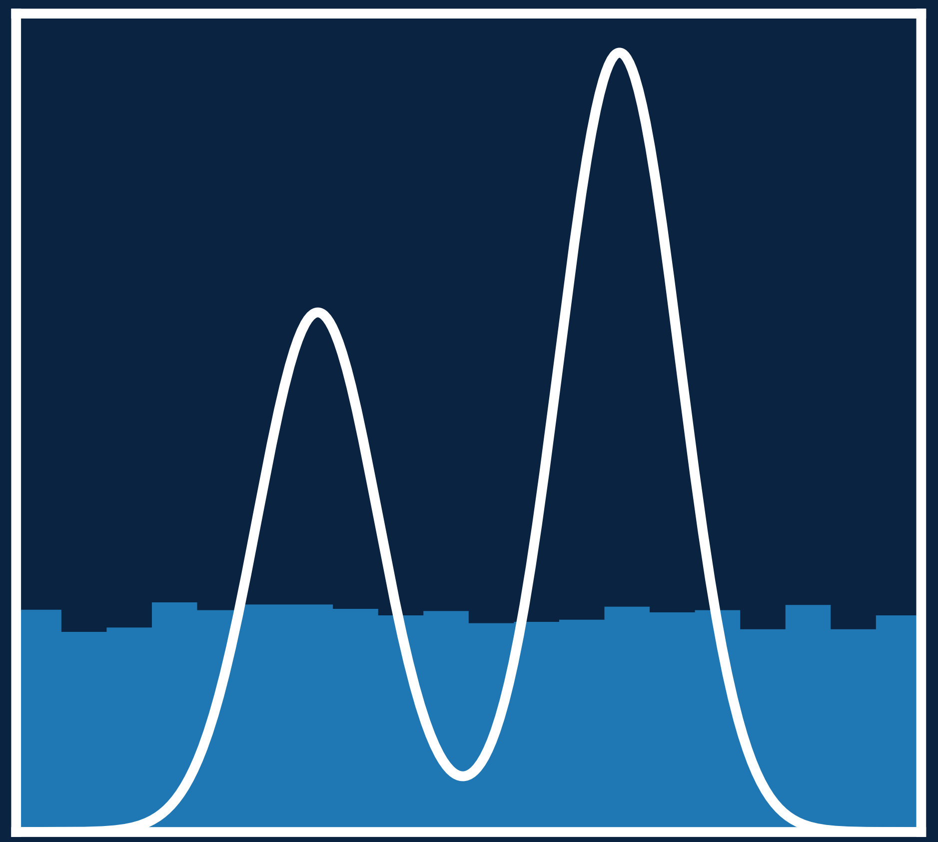
Importance sampling
BDT [1707.00028], NN [1810.11509, 2009.07819]
NF [2001.05486, 2001.05478, 2001.10028, 2005.12719, 2112.09145, 2212.06172, 2311.01548, 2401.09069]
Chili [2302.10449]

Surrogate regression
Full weight [2109.11964],
Matrix element [1912.11055, 2002.07516, 2006.16273, 2106.09474, 2107.06625, 2109.11964, 2206.14831, 2301.13562, 2302.04005, 2306.07726]

Event generation

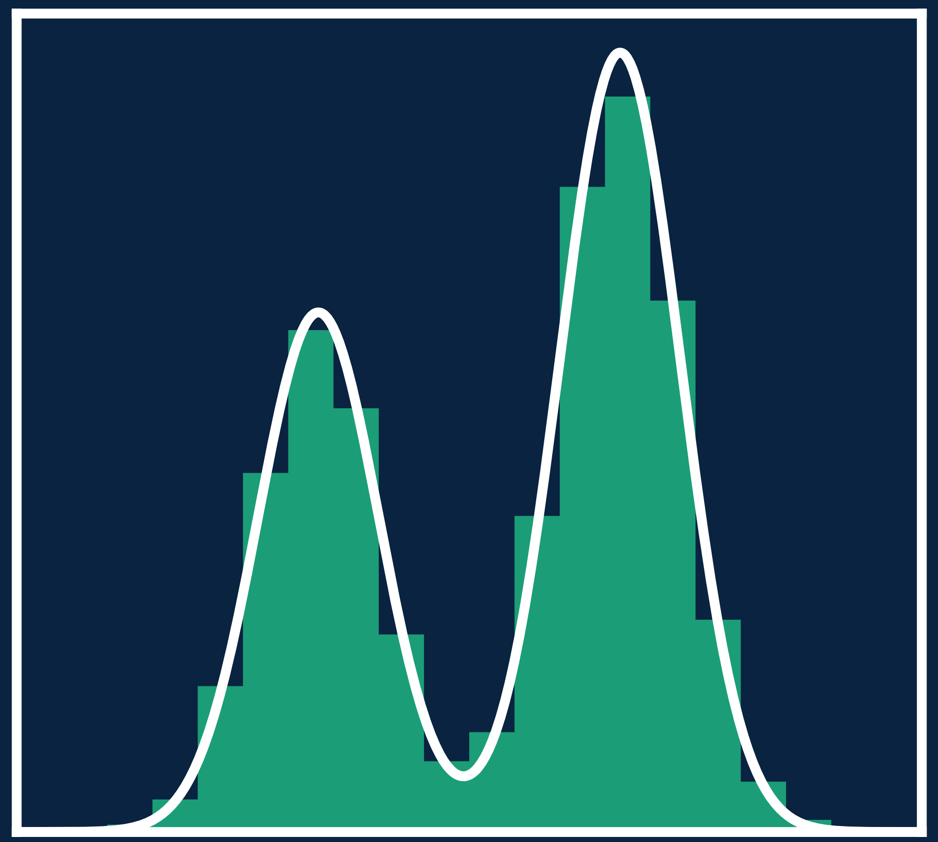
Calculate (differential) cross sections

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



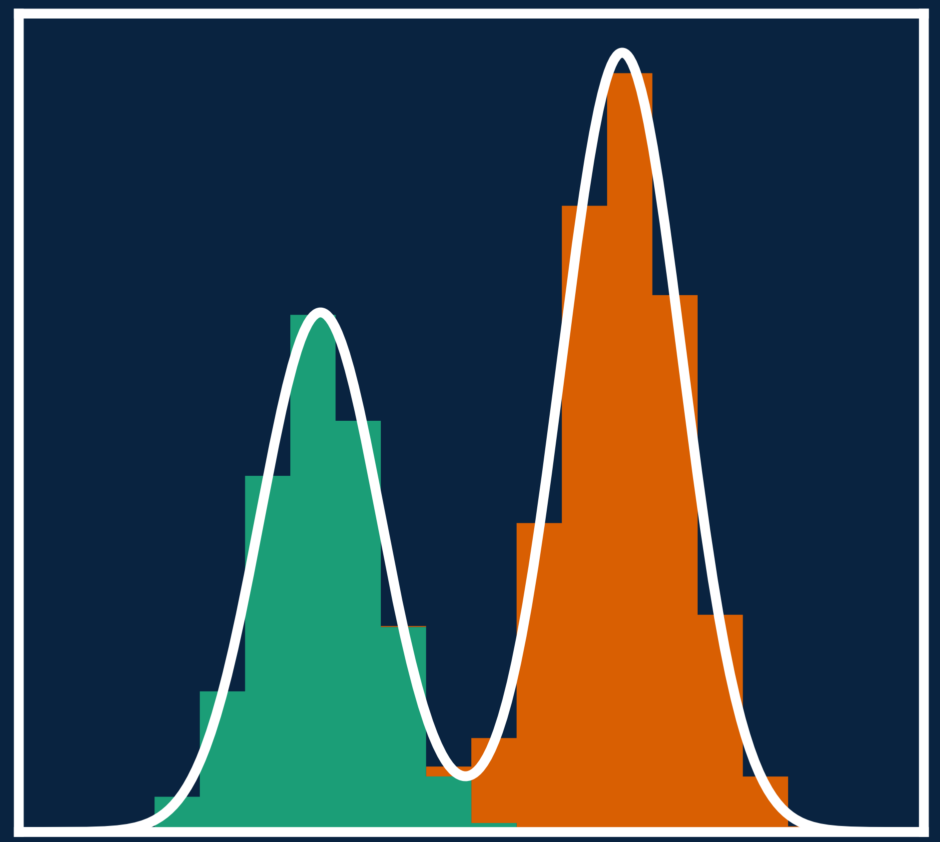
Flat sampling:
inefficient

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



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

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

Sum over channels

MadGraph: build channels from Feynman diagrams

Integrand

MadGraph: $d\sigma / dx$

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

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)

MadNIS

Neural Importance Sampling

Heimel, Huetsch, Maltoni, Mattelaer, Plehn, RW [2311.01548]

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

MadNIS — Basic functionality

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

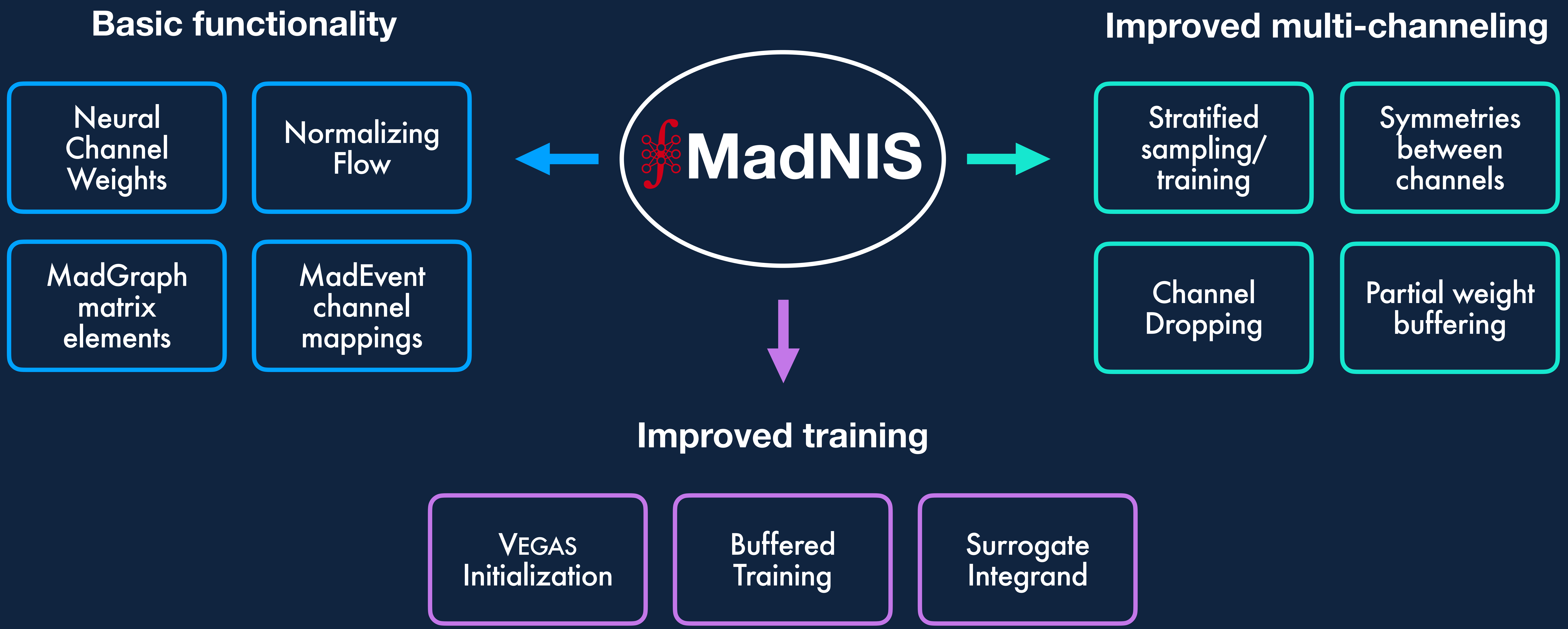
Use physics knowledge to construct channel and mappings

Normalizing flow to refine channel mappings

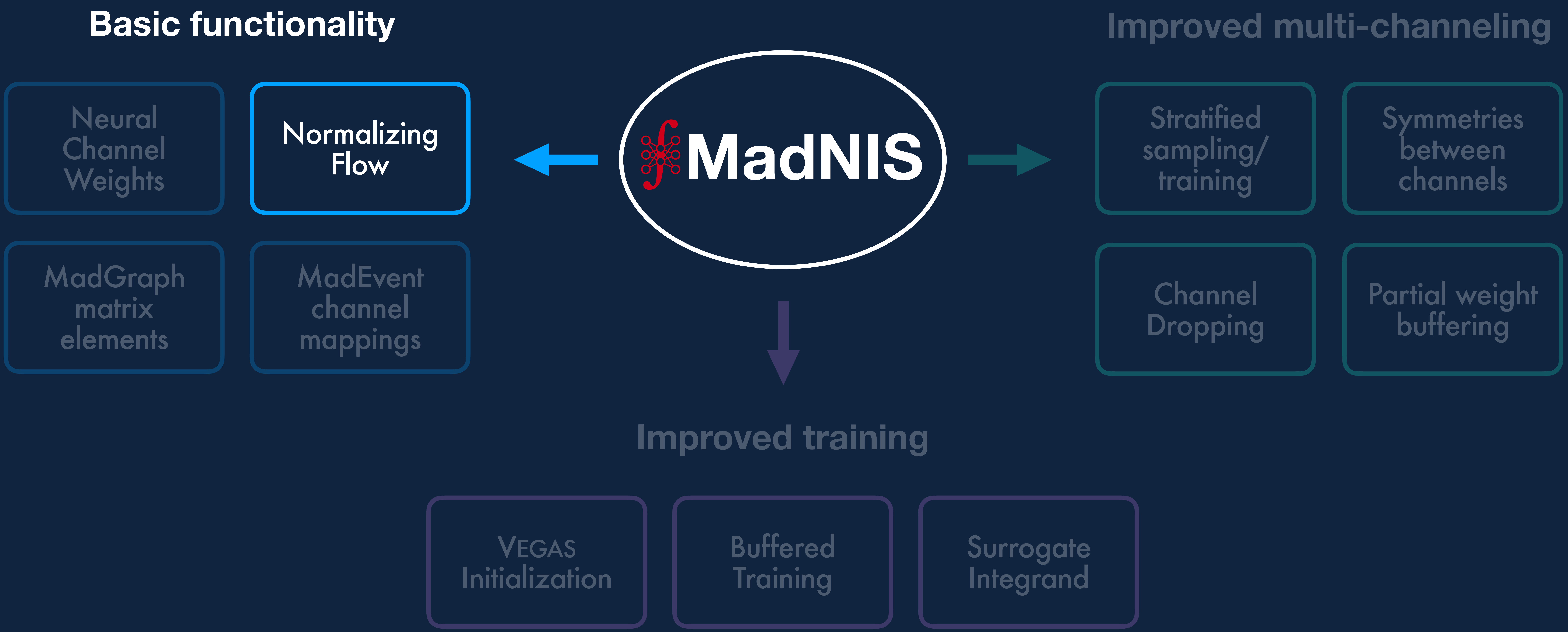
Fully connected network to refine channel weights

Update simultanously with variance as loss function

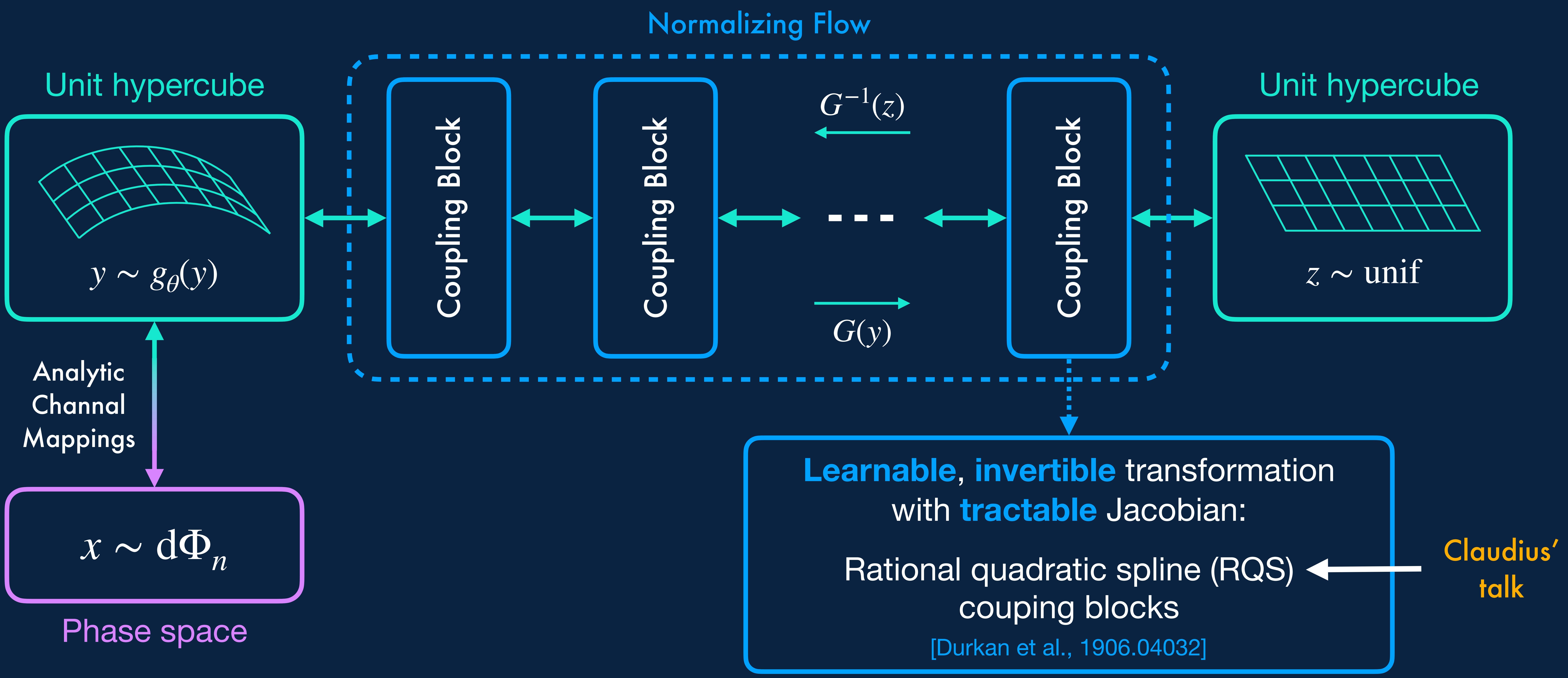
MadNIS — Overview



MadNIS — Overview

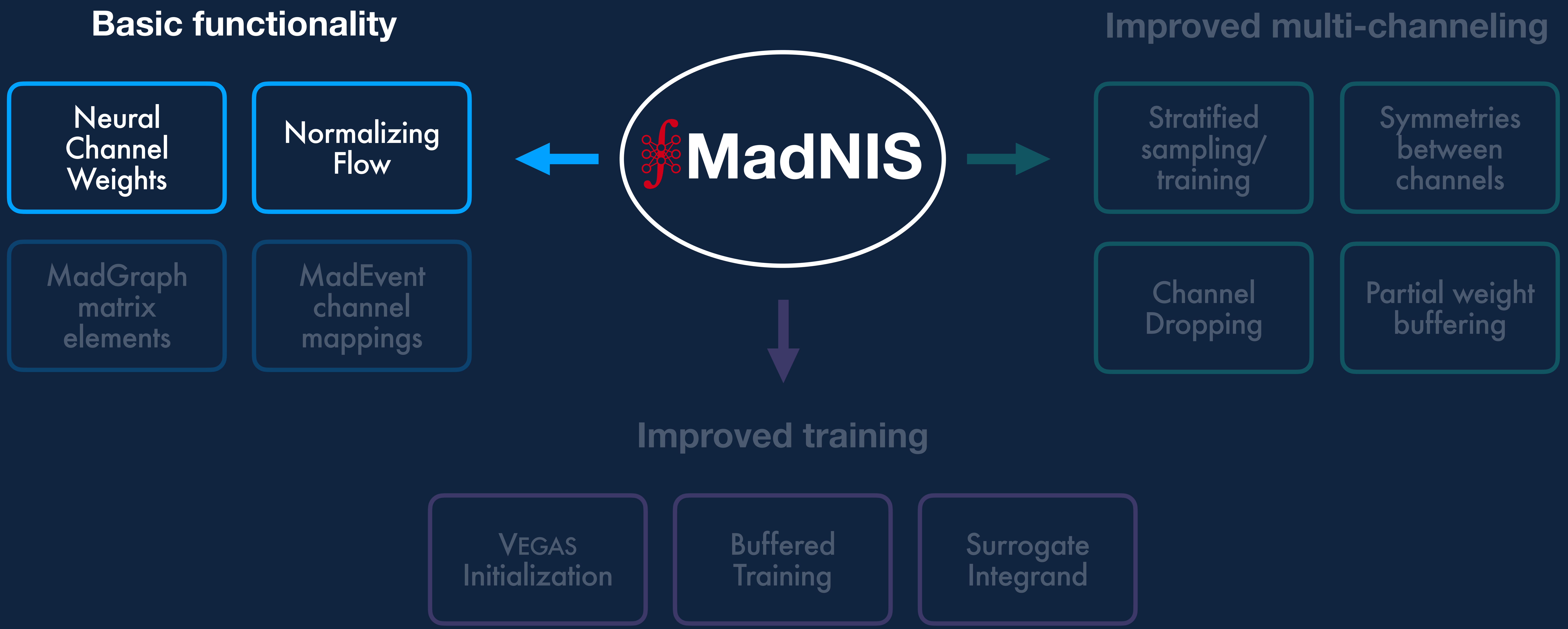


Neural importance sampling

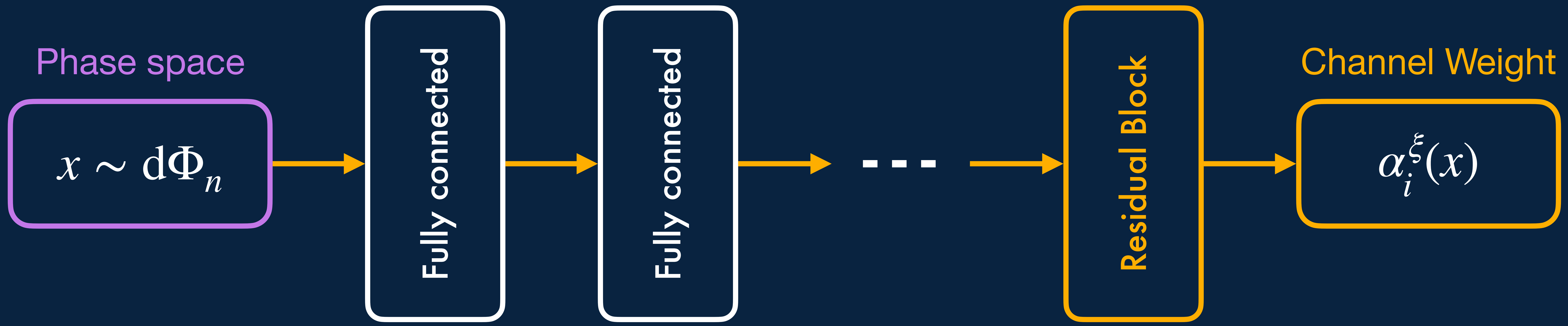


Flows for NIS: [Gao et al, 2001.05486] [Gao et al, 2001.10028] [Bothmann et al, 2001.05478] [Winterhalder et al, 2112.09145]

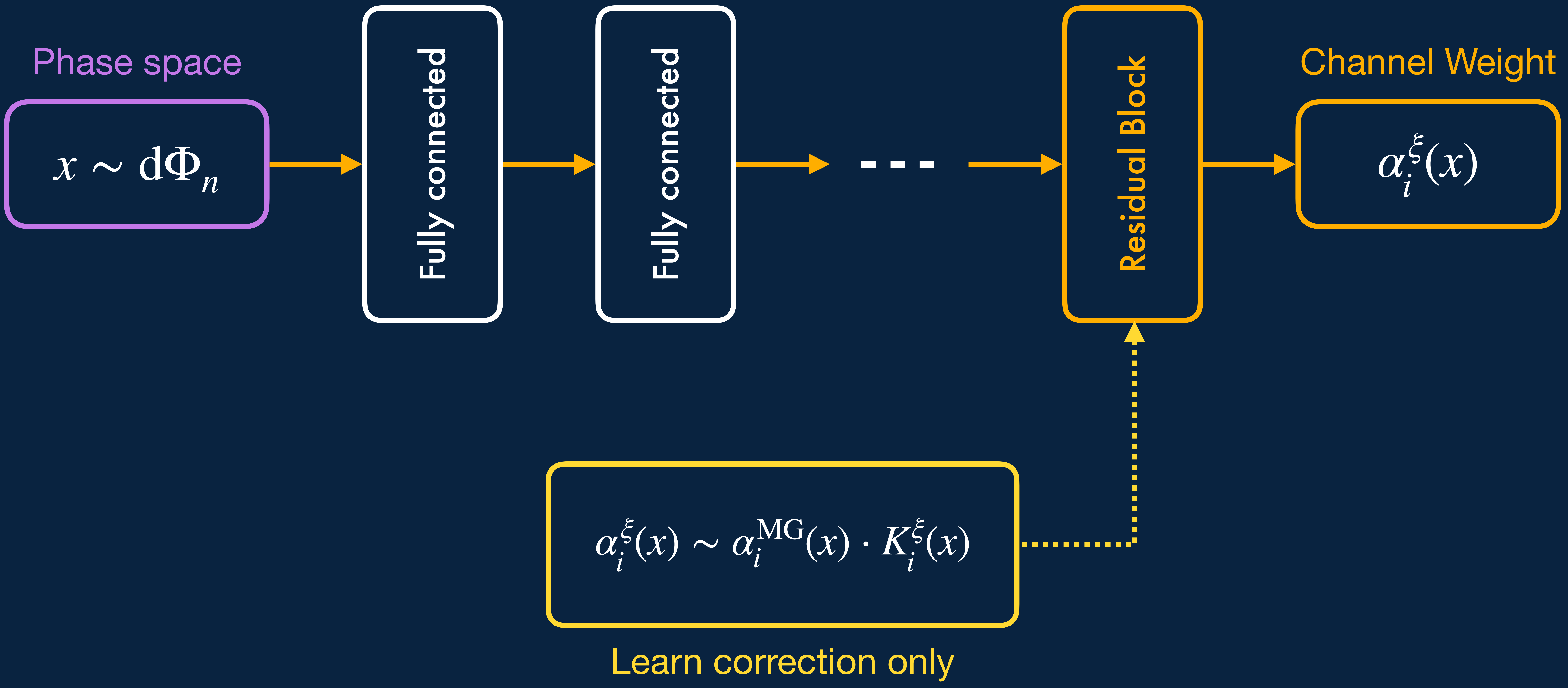
MadNIS — Overview



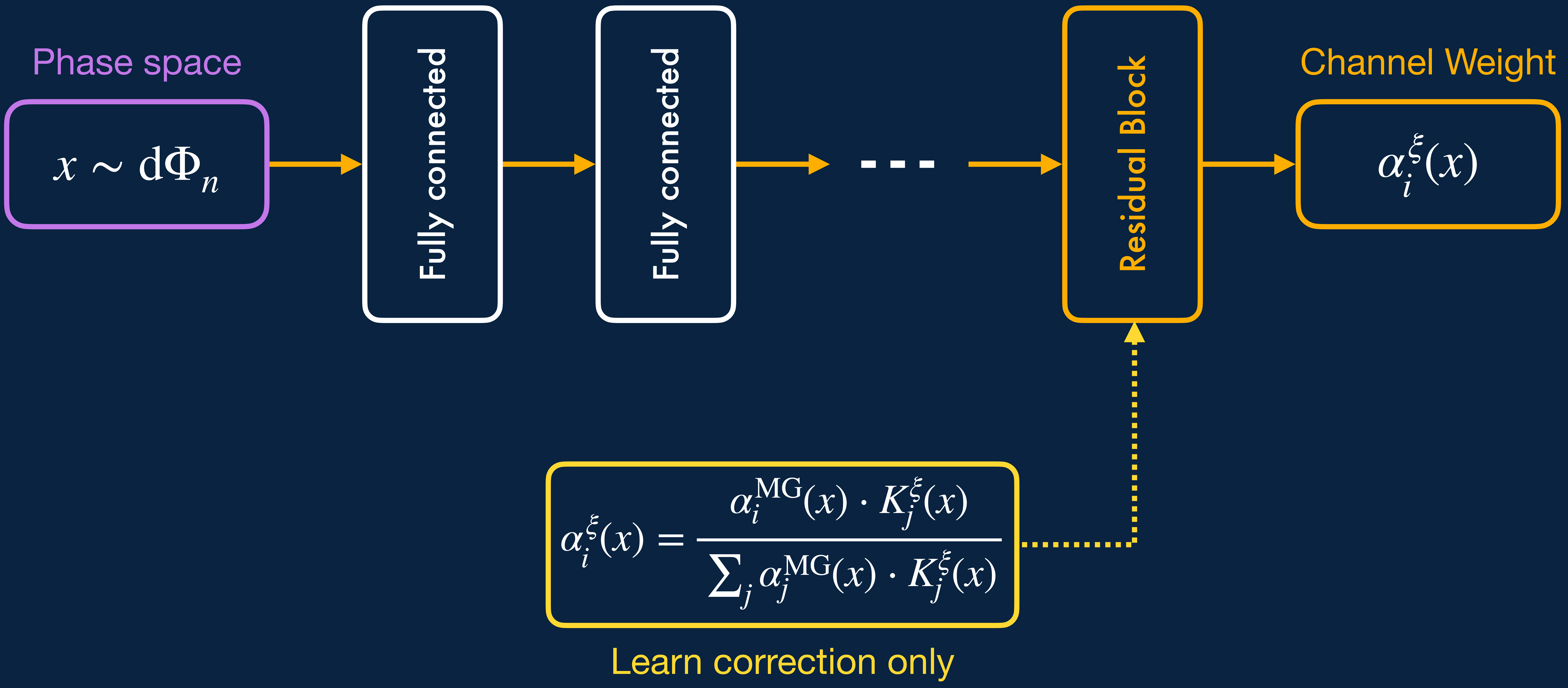
Neural Channel Weights



Neural Channel Weights



Neural Channel Weights



MadNIS — Overview

Basic functionality

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

MadNIS

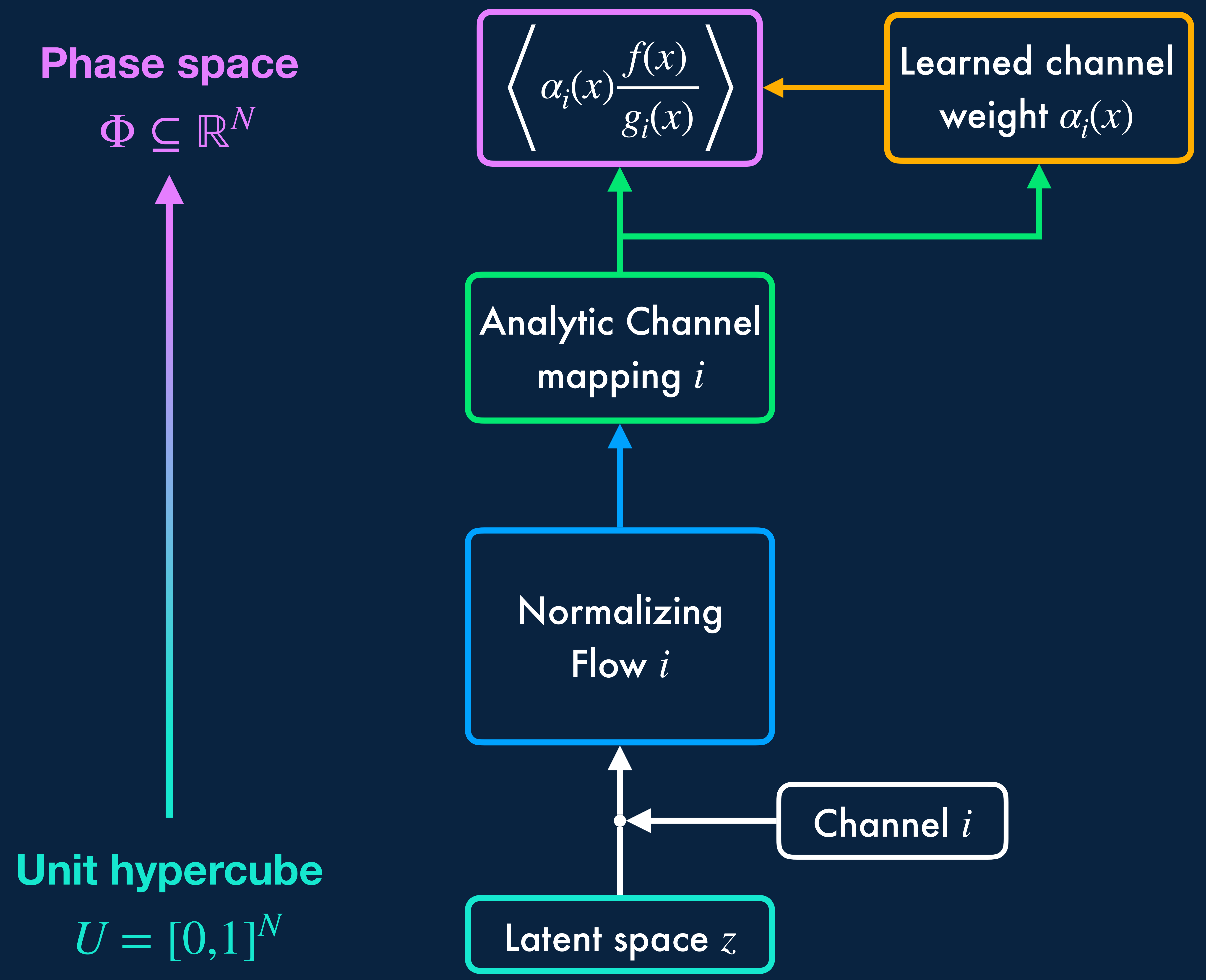
Improved multi-channeling

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

Improved training

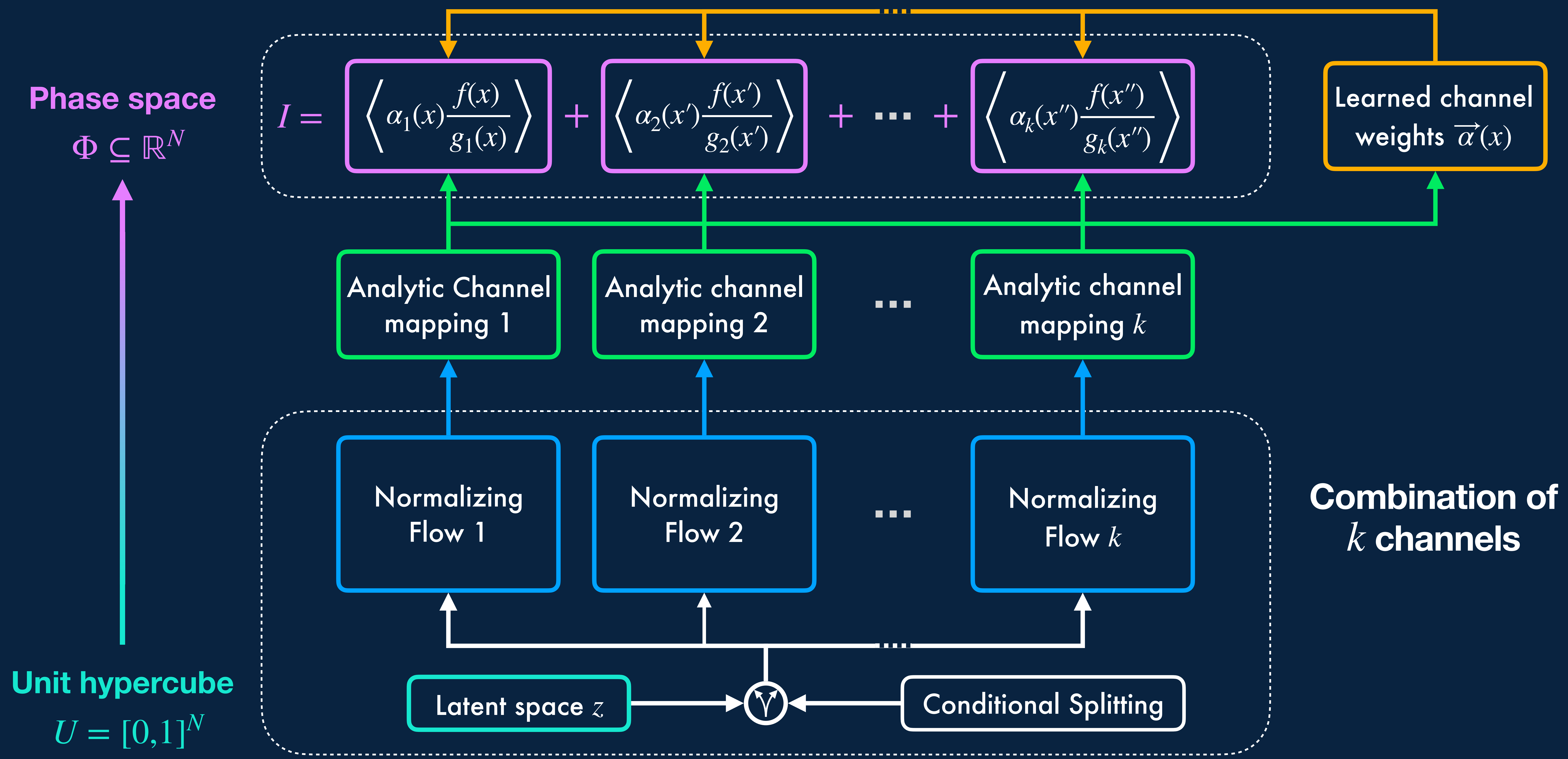
- VEGAS Initialization
- Buffered Training
- Surrogate Integrand

MadNIS — Basic functionality



Single channel i

MadNIS — Basic functionality



Loss function

Minimize total variance

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

$$\sigma_i^2 = \left\langle \alpha_i^2(x) \frac{f^2(x)}{g_i^2(x)} \right\rangle_{x \sim g_i(x)} - \left\langle \alpha_i(x) \frac{f(x)}{g_i(x)} \right\rangle_{x \sim g_i(x)}^2$$

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

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

$$\sigma_i^2 = \left\langle \alpha_i^2(x) \frac{f^2(x)}{g_i^2(x)} \right\rangle_{x \sim g_i(x)} - \left\langle \alpha_i(x) \frac{f(x)}{g_i(x)} \right\rangle_{x \sim g_i(x)}^2$$

Total variance depends on N_i

↓

affects optimal $\alpha_i(x)$

↓

use stratified sampling

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

↓ ↓

MadNIS loss function

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

MadNIS — Overview

MadNIS

Basic functionality

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

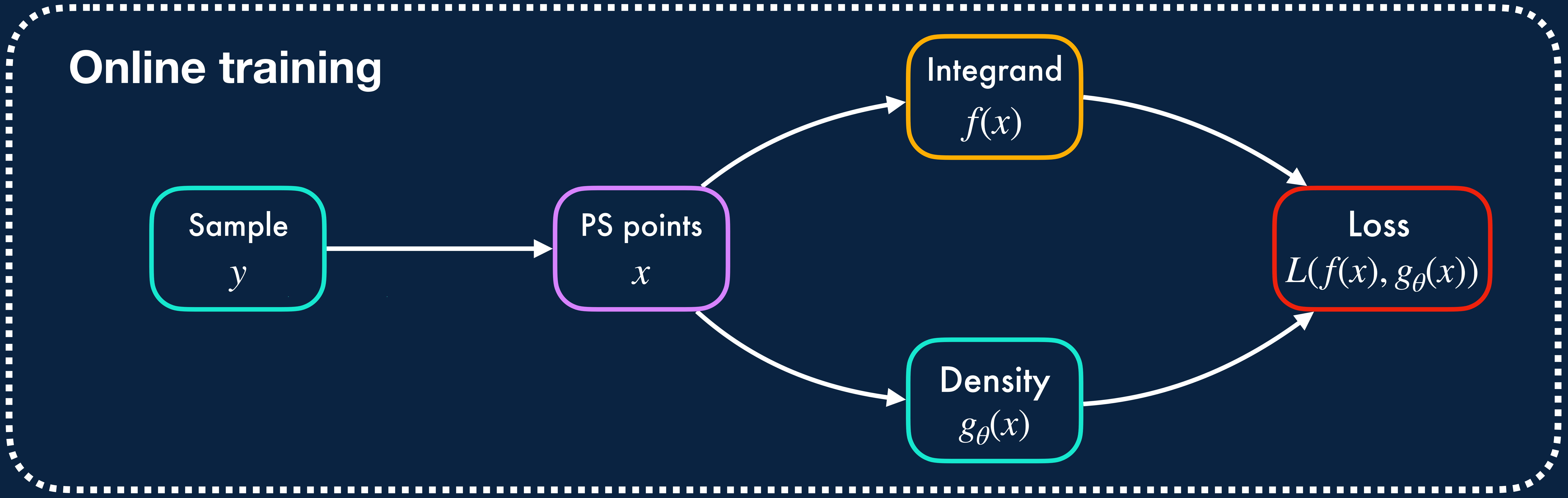
Improved multi-channeling

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

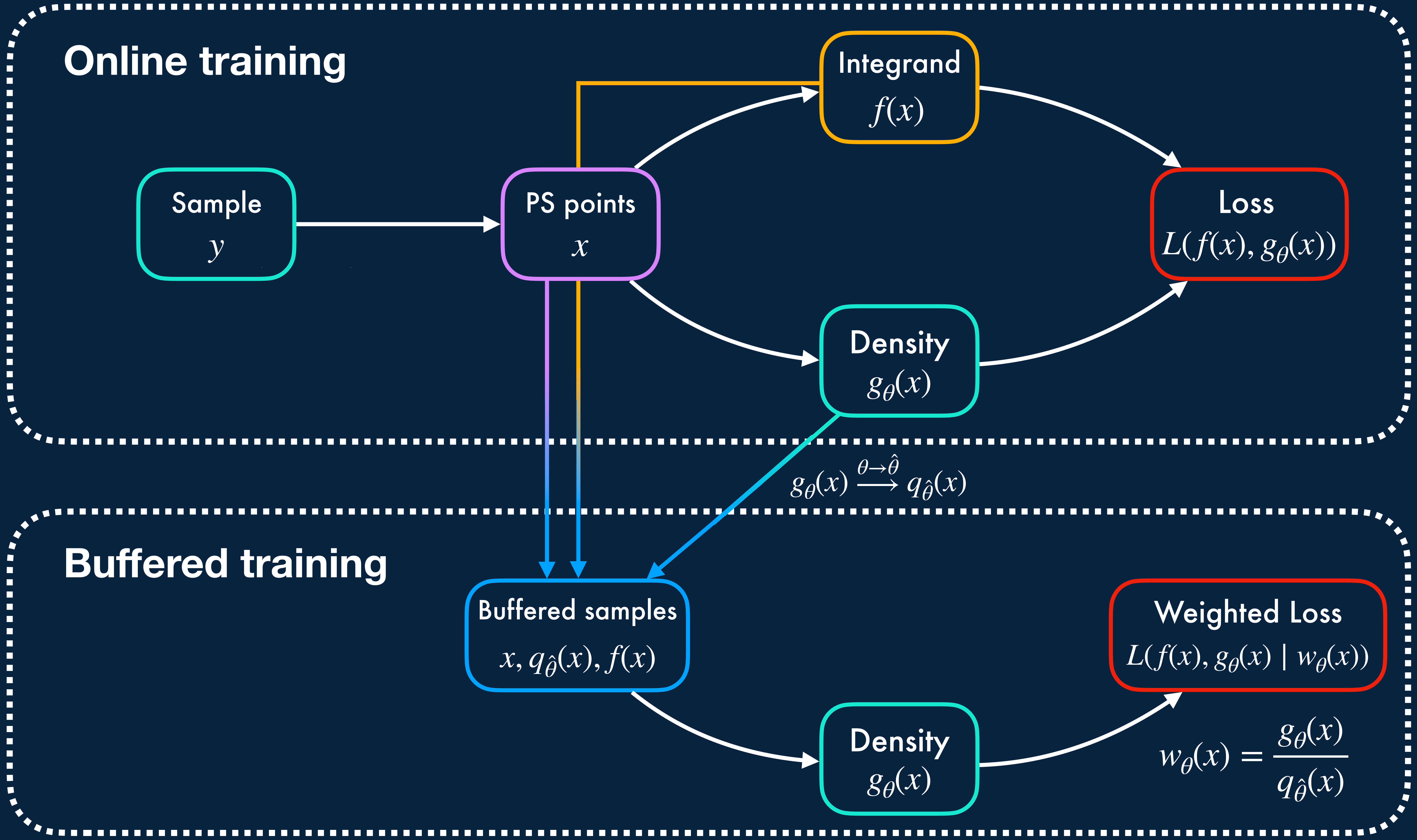
Improved training

- VEGAS Initialization
- **Buffered Training**
- Surrogate Integrand

Buffered training



Buffered training



Buffered training

Training algorithm

generate new samples, train on them, save samples

↓

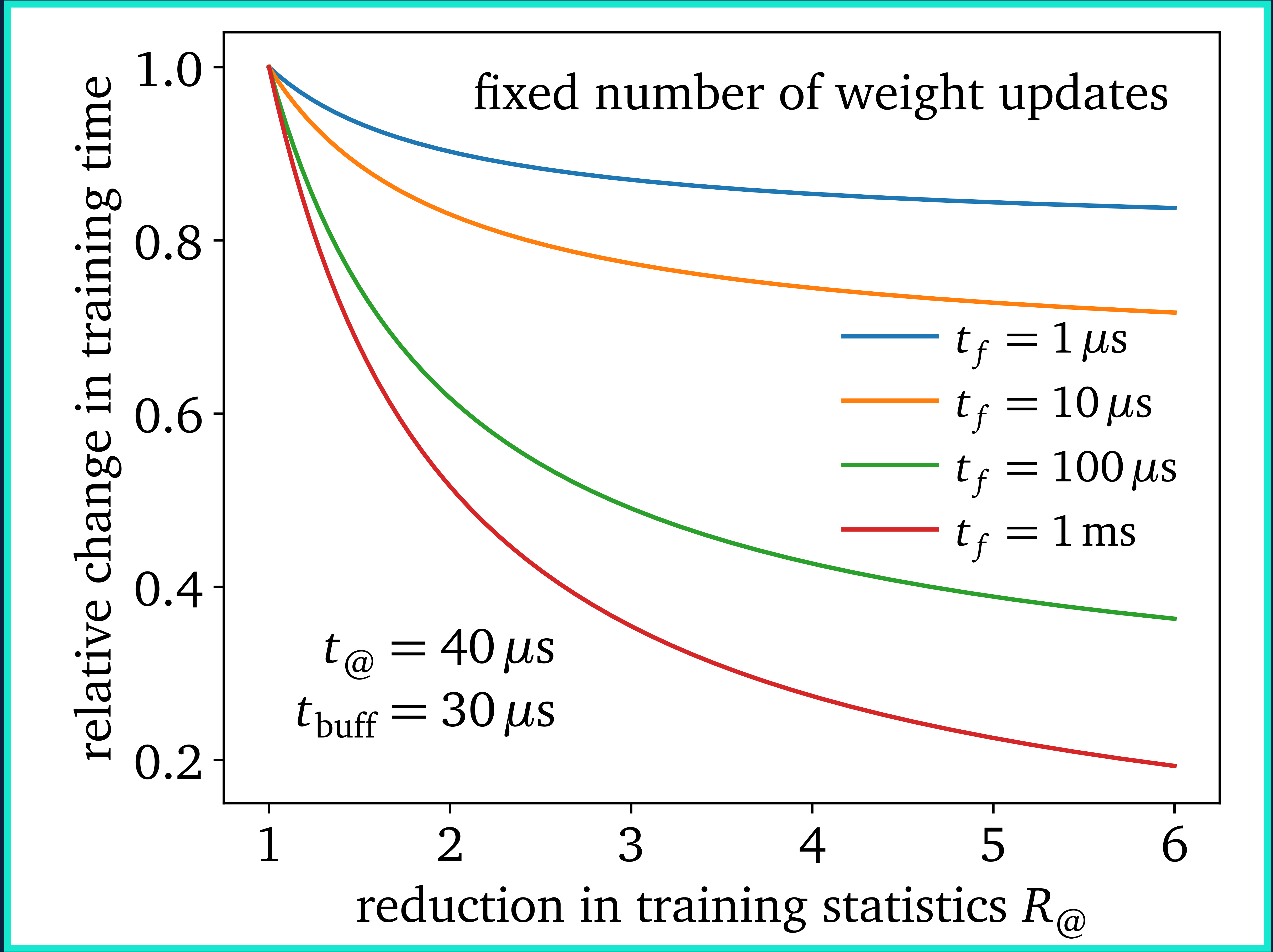
train on saved samples n times

↓

repeat

Reduction in training statistics by

$$R_@ = n + 1$$



MadNIS — 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:

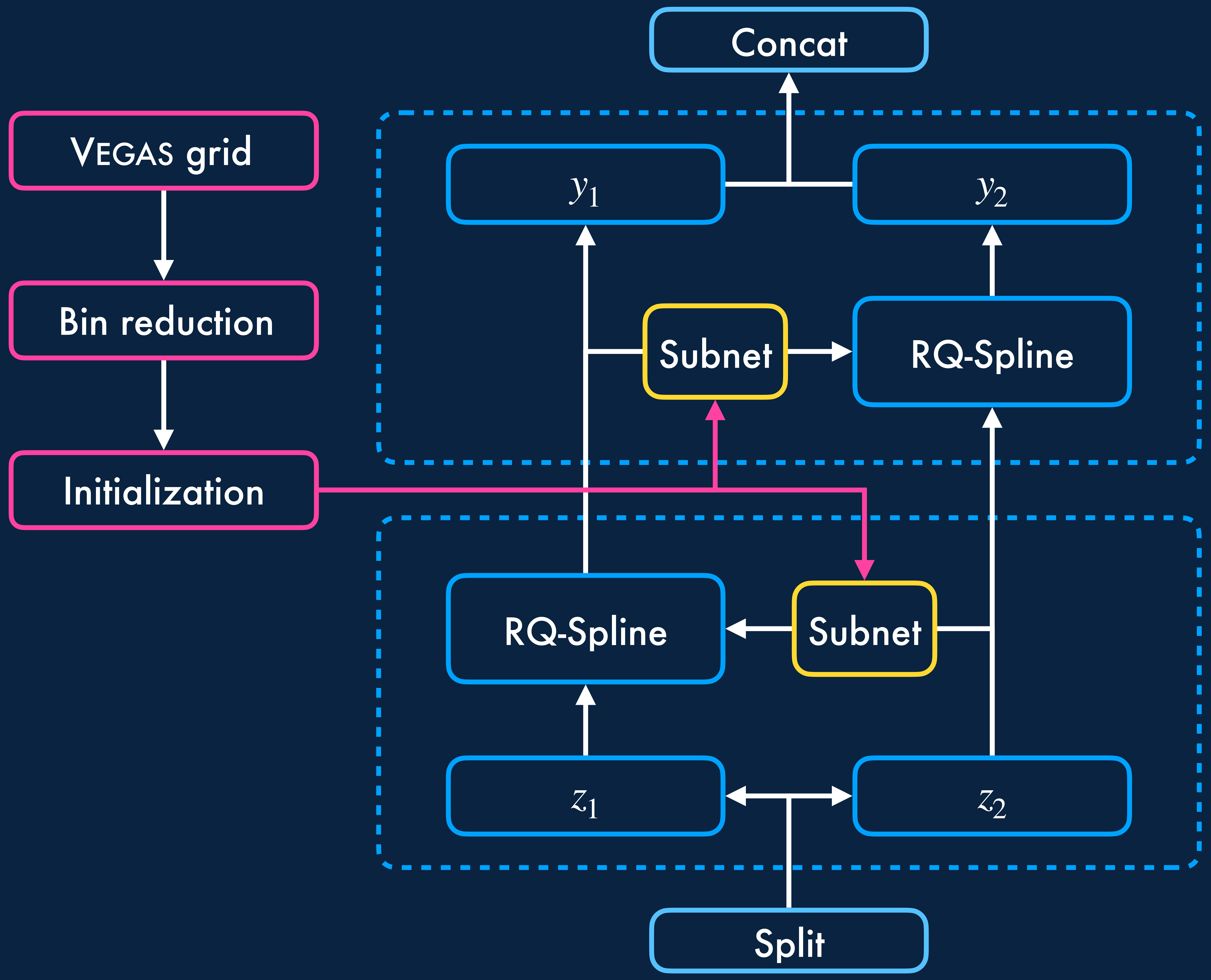
Pre-trained VEGAS grid as starting point for flow training

VEGAS initialization

	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

Combine advantages:

Pre-trained VEGAS grid as starting point for flow training



MadNIS — 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

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

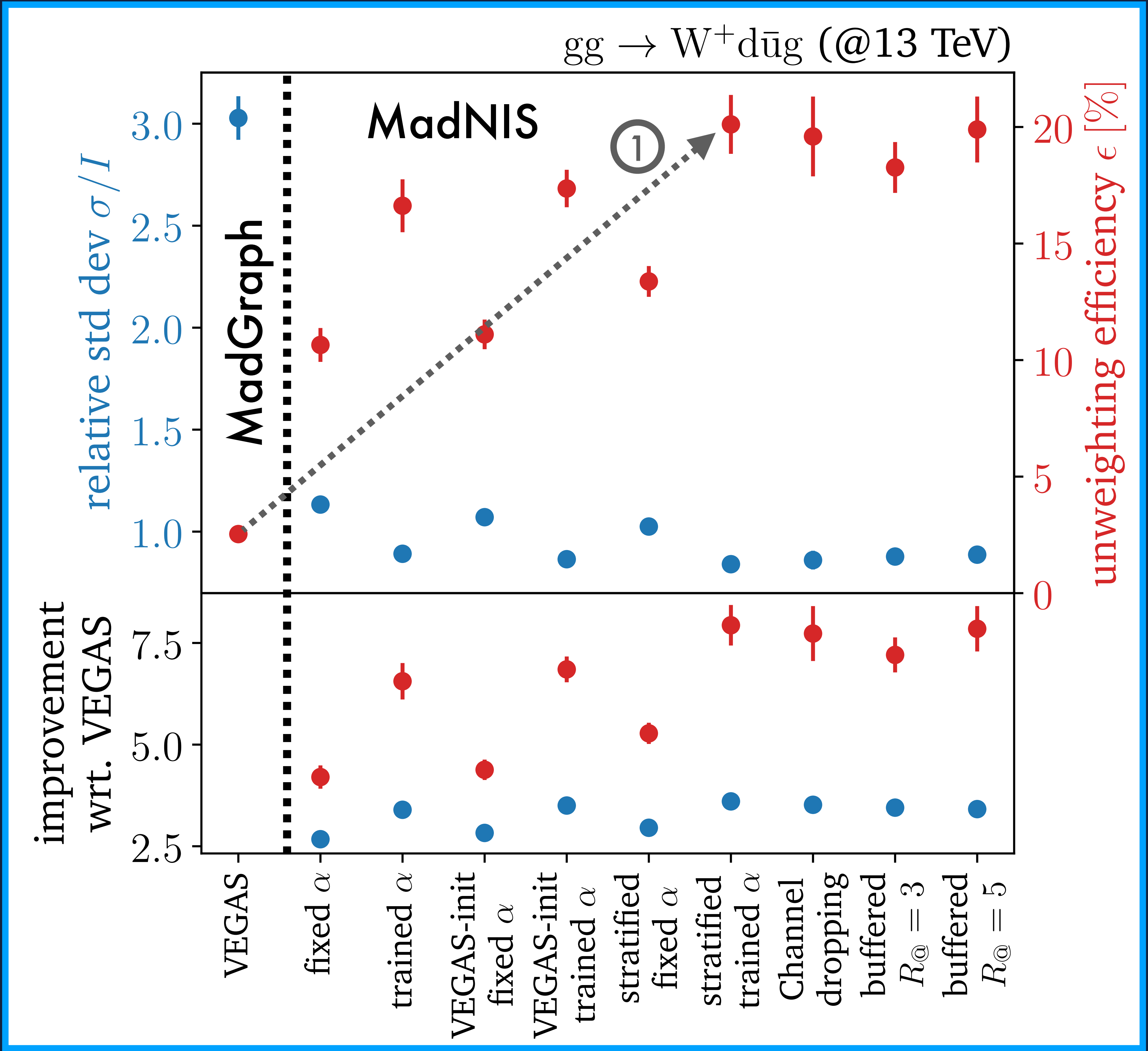
MadNIS often **reduces contribution** of some channels to total integral

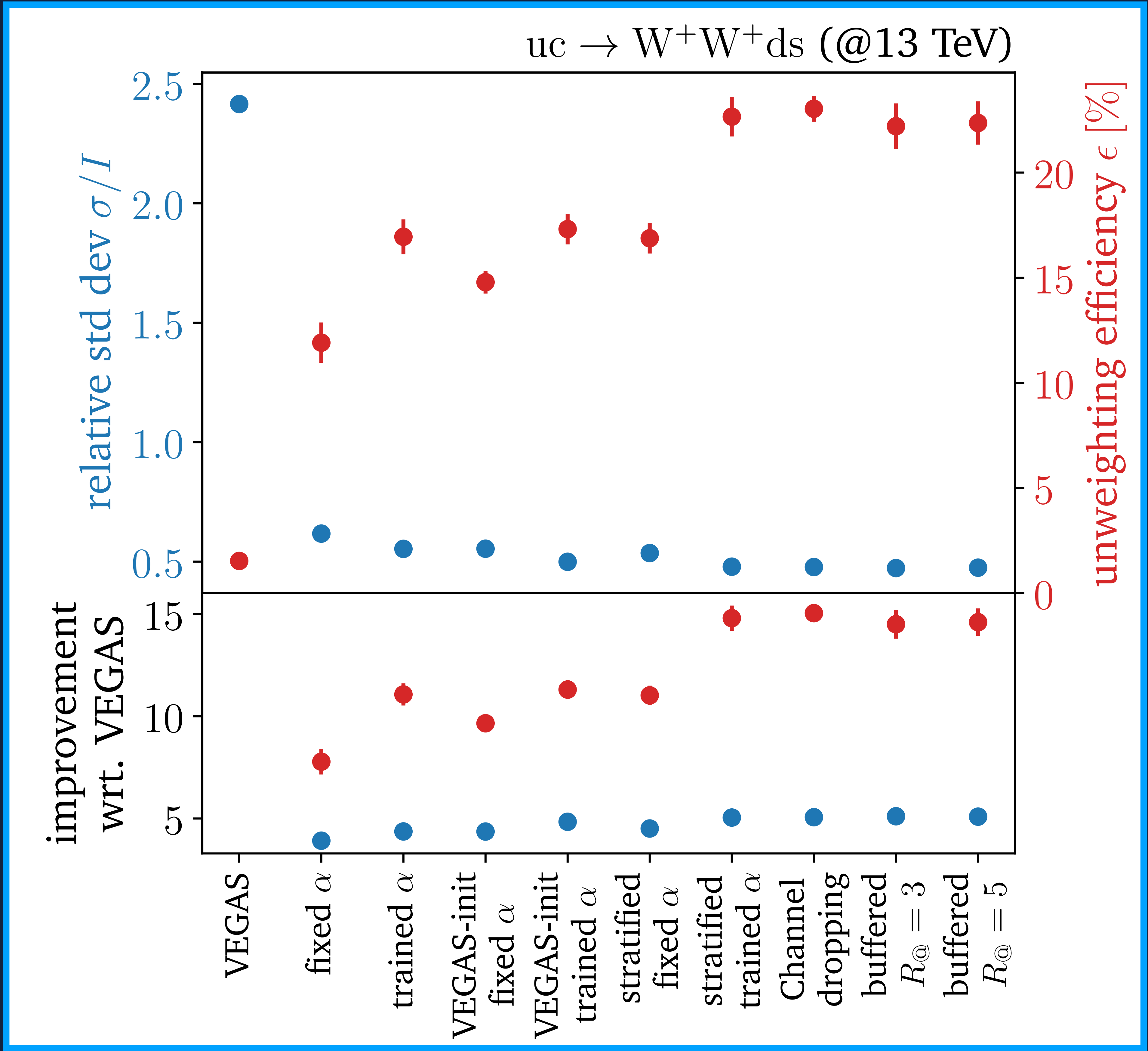
↓

remove these channels from the **training** completely

Reduced complexity
Improved stability

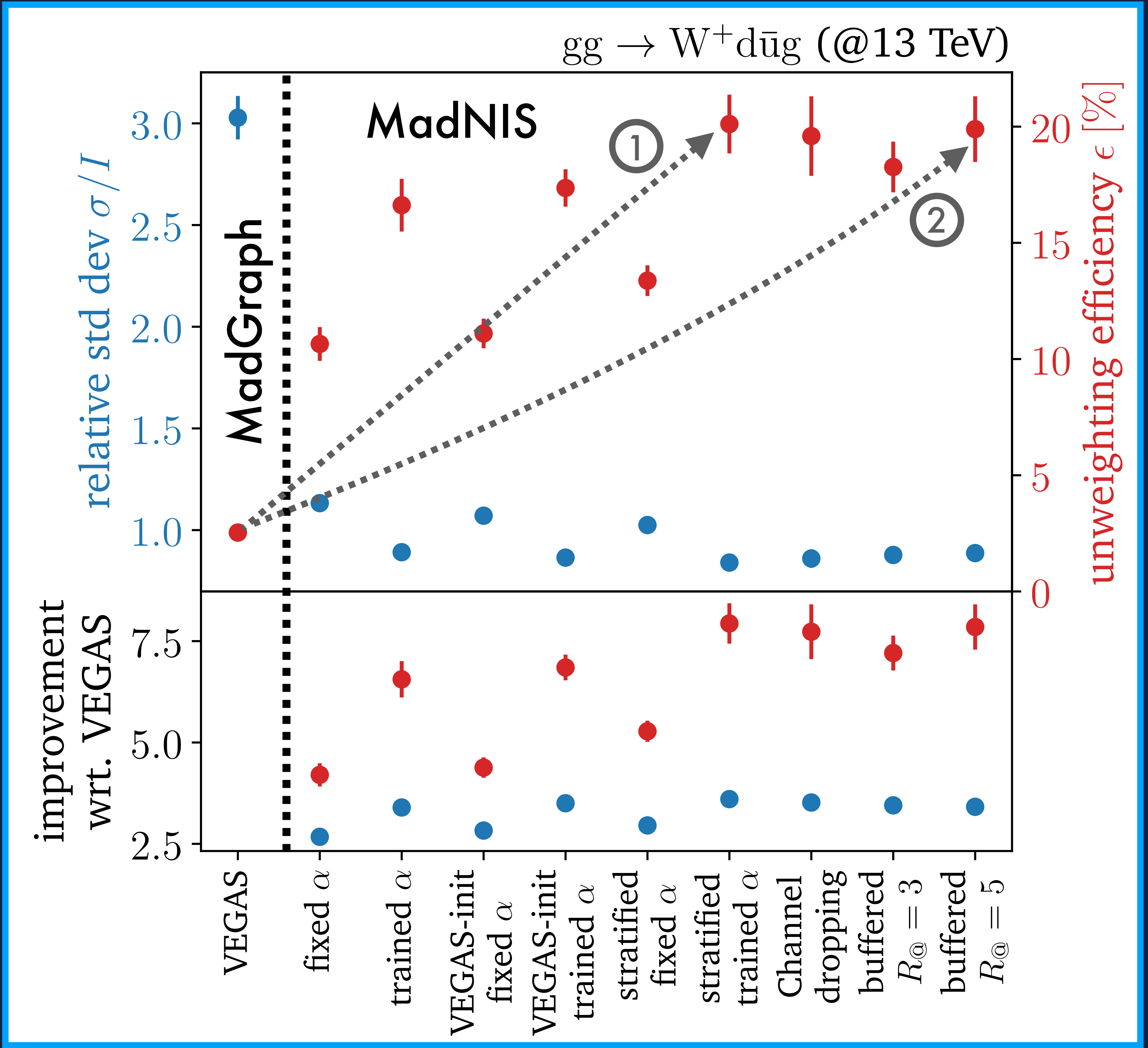
LHC processes

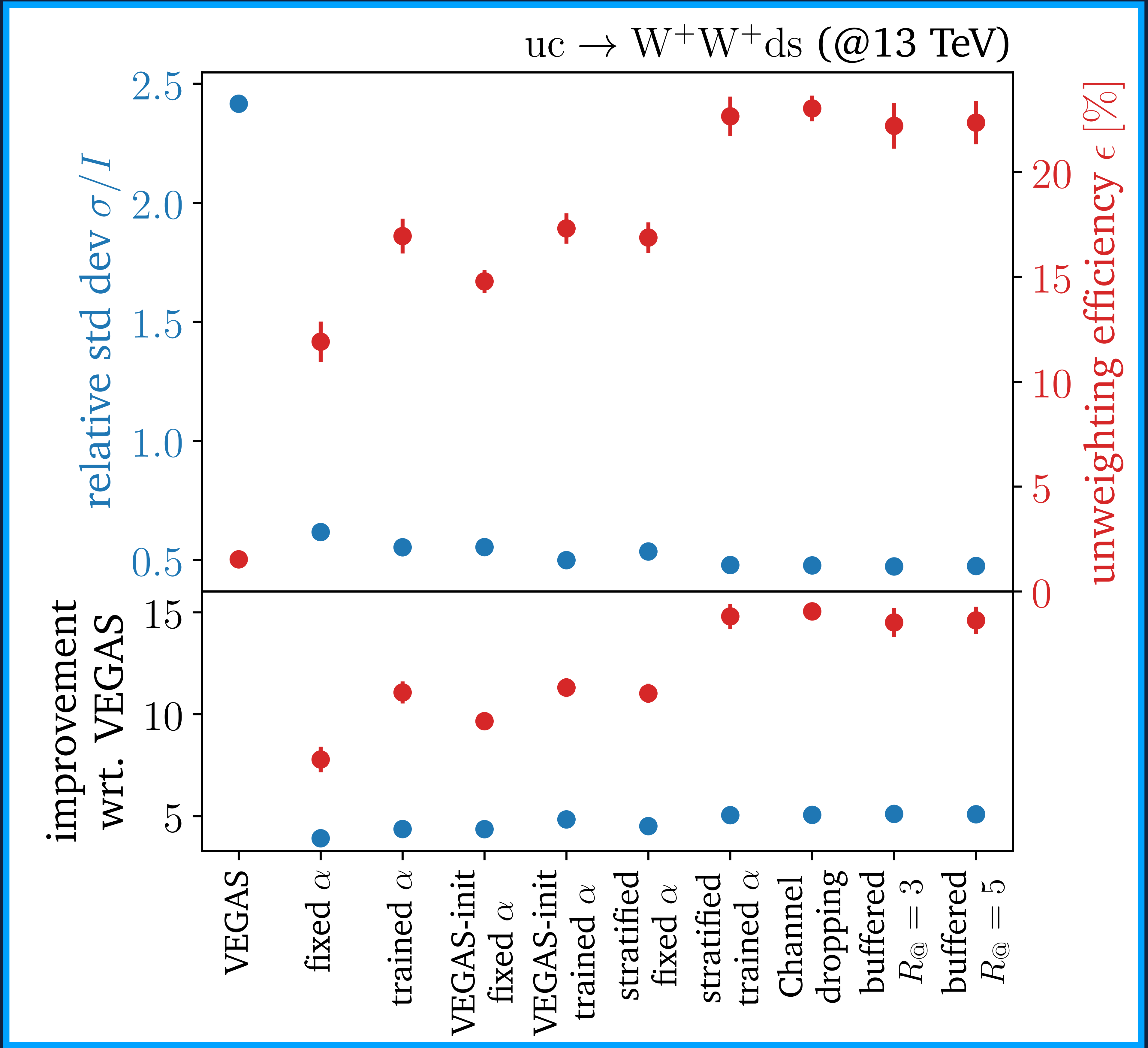




1. excellent results with all improvements

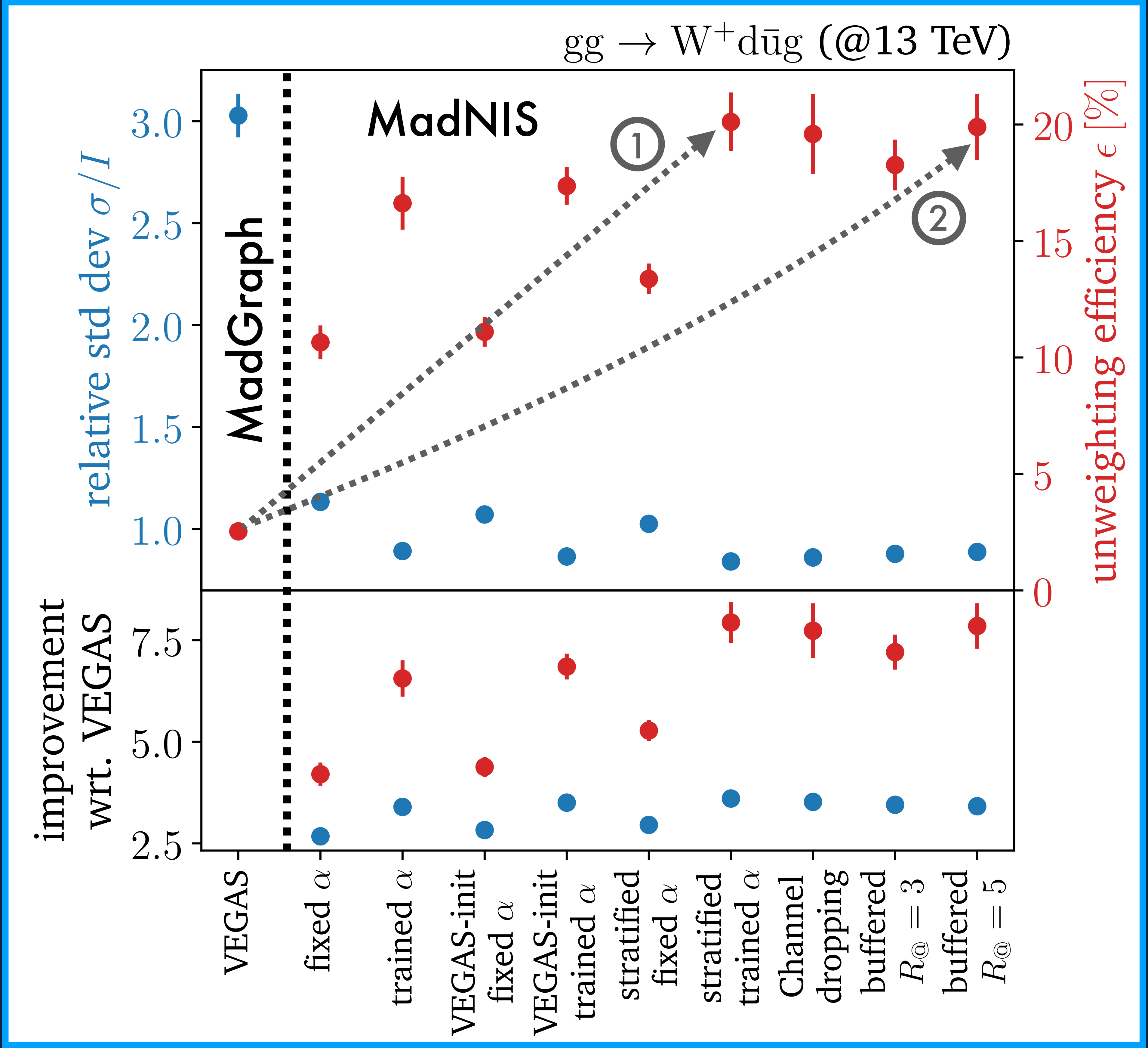
LHC processes

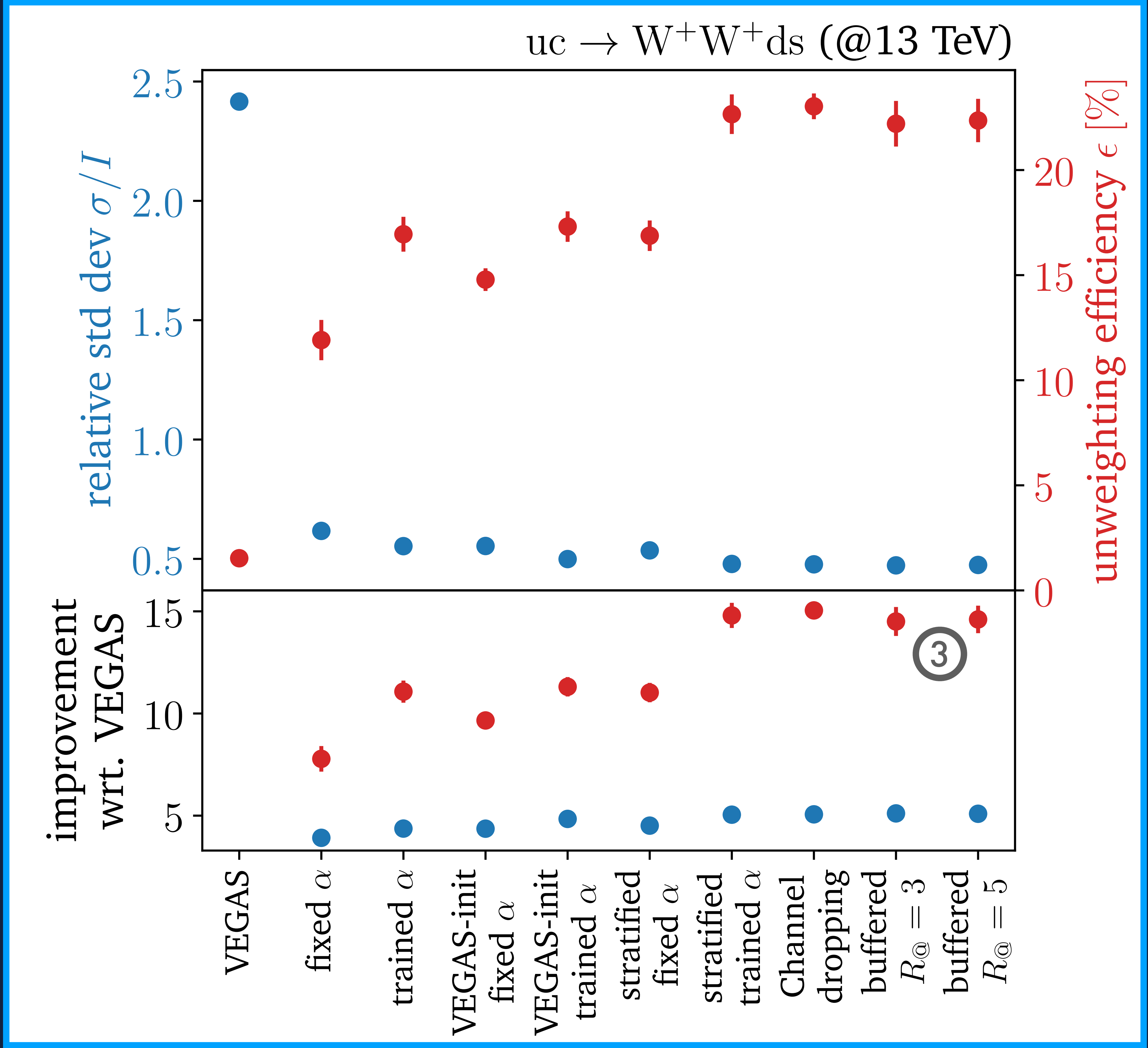




1. excellent results with all improvements
2. same performance with buffered training

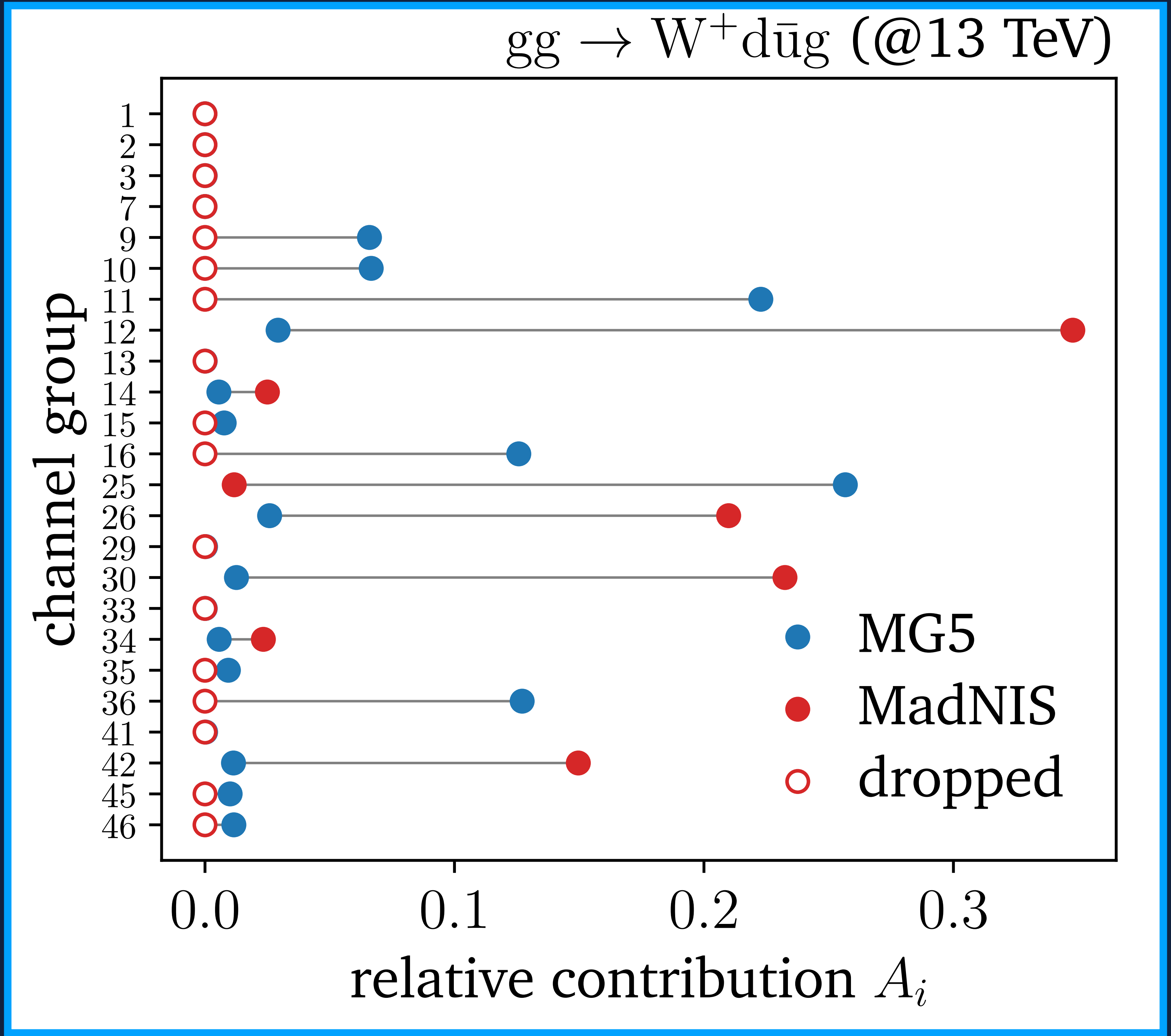
LHC processes

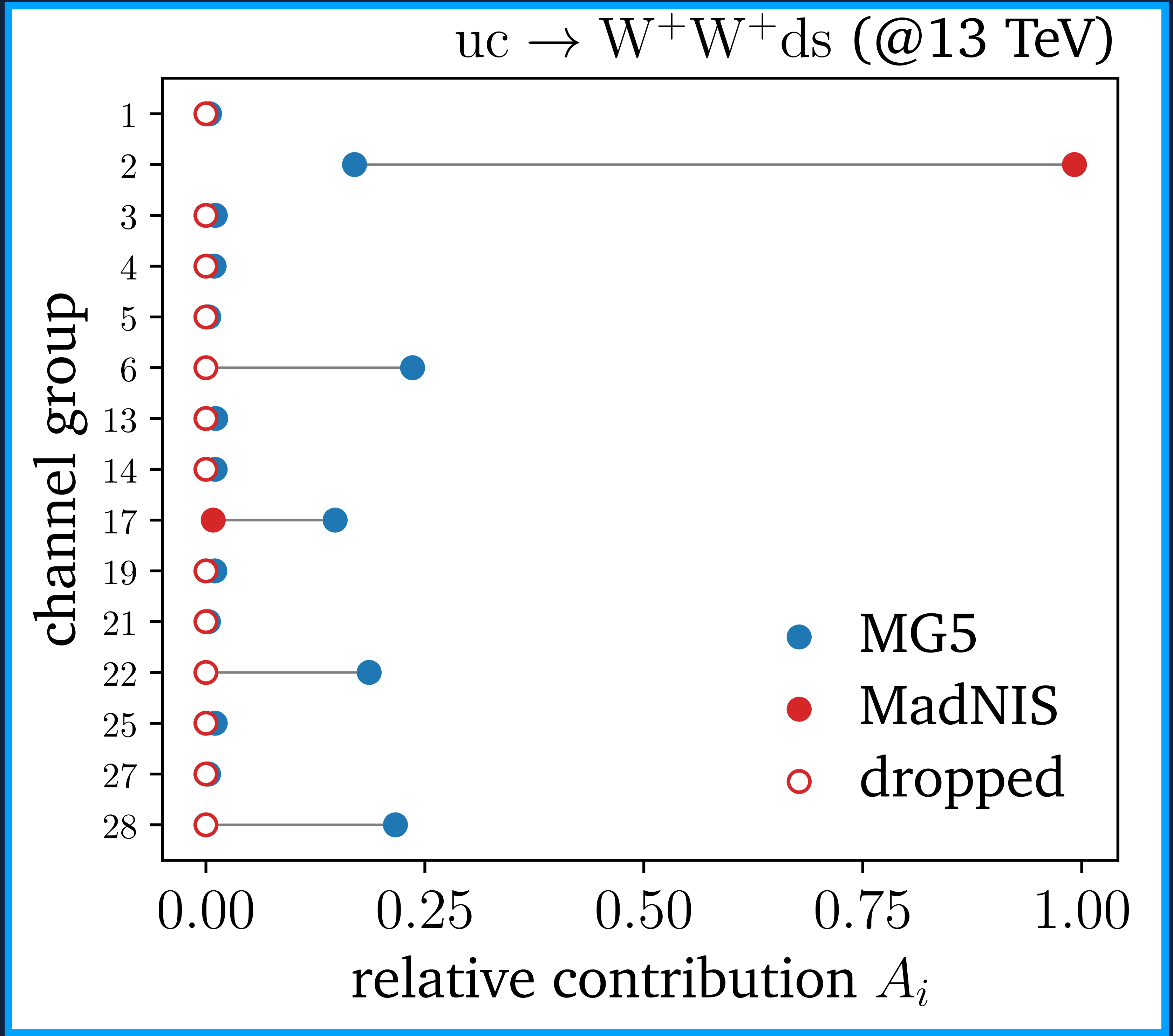




1. excellent results with all improvements
2. same performance with buffered training
3. Larger improvements for processes with large interference terms

Learned channel weights





In MadNIS many channels **are zero**

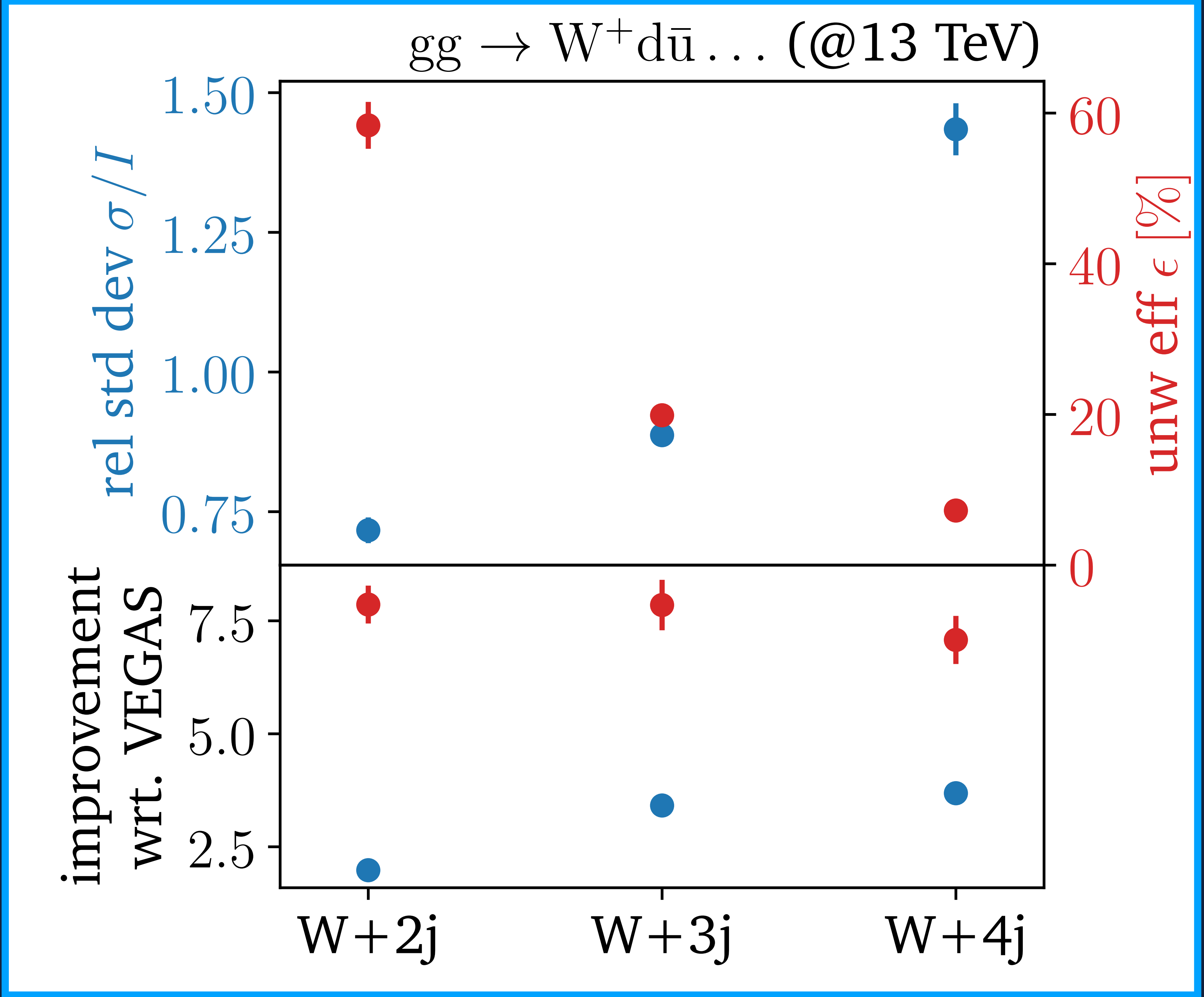
↓

droppig channels

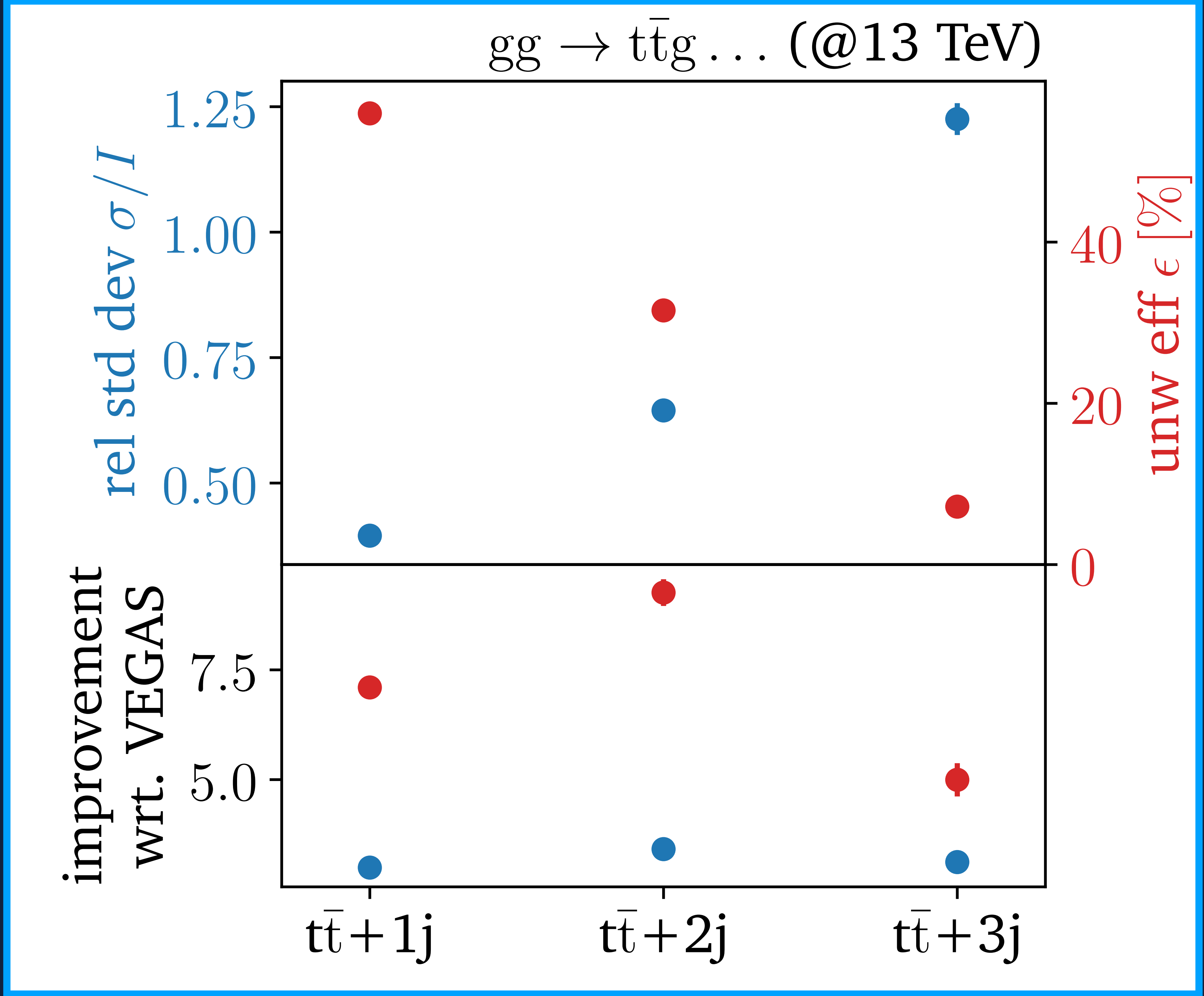
↓

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

Outlook

HEP ML Living Review

Search

GitHub 246 78

Home Recent About Contribute Resources Cite Us

A Living Review of Machine Learning for Particle Physics

Modern machine learning techniques, including deep learning, is rapidly being applied, adapted, and developed for high energy physics. The goal of this document is to provide a nearly comprehensive list of citations for those developing and applying these approaches to experimental, phenomenological, or theoretical analyses. As a living document, it will be updated as often as possible to incorporate the latest developments. A list of proper (unchanging) reviews can be found within. Papers are grouped into a small set of topics to be as useful as possible. Suggestions are most welcome.

download review | GitHub

Expand all sections | Collapse all sections

Reviews

- **Modern reviews**
- **Specialized reviews**
- **Classical papers**
- **Datasets**

Table of contents

HEPML

Backup

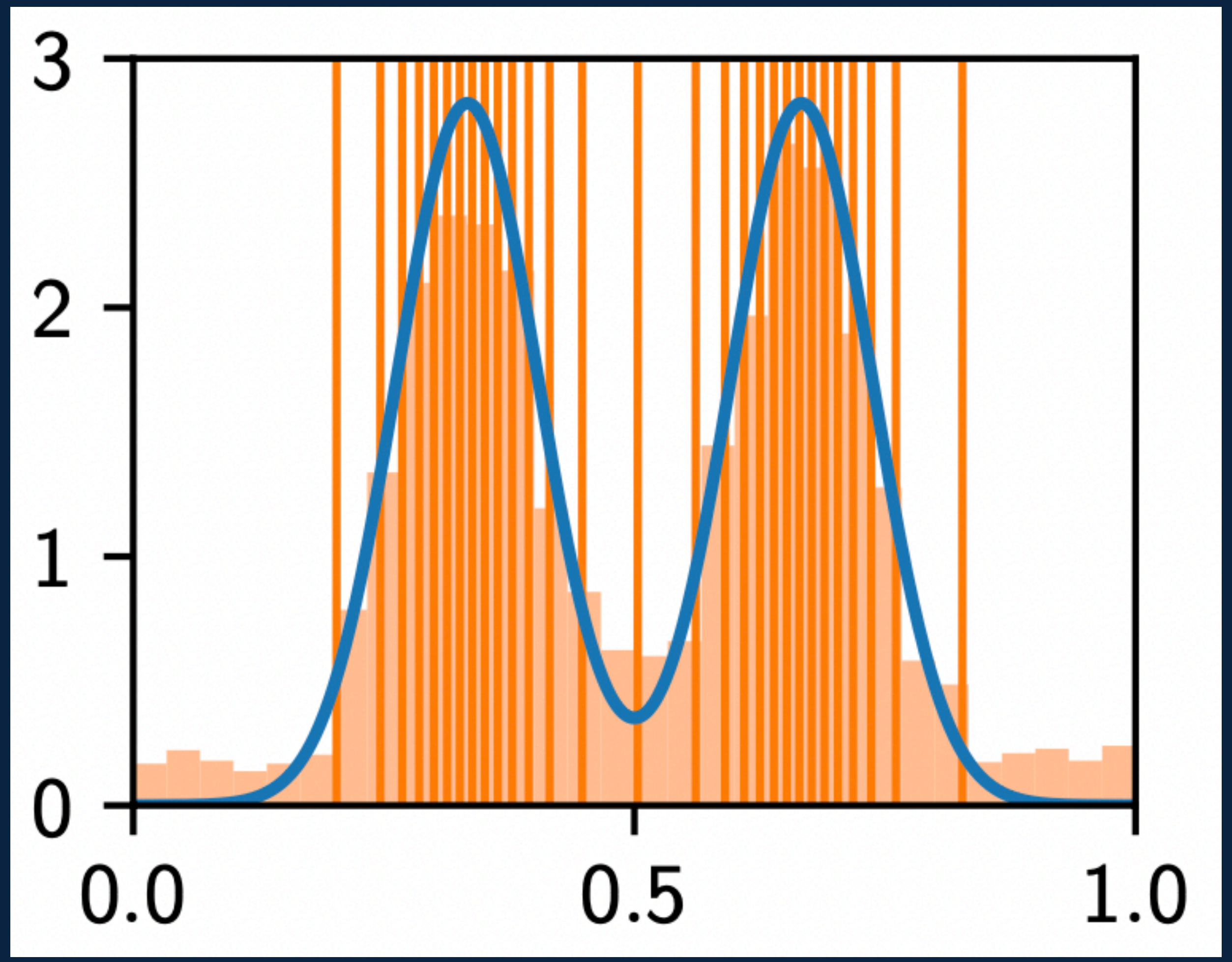
Appendix

Importance sampling — VEGAS

Factorize probability

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

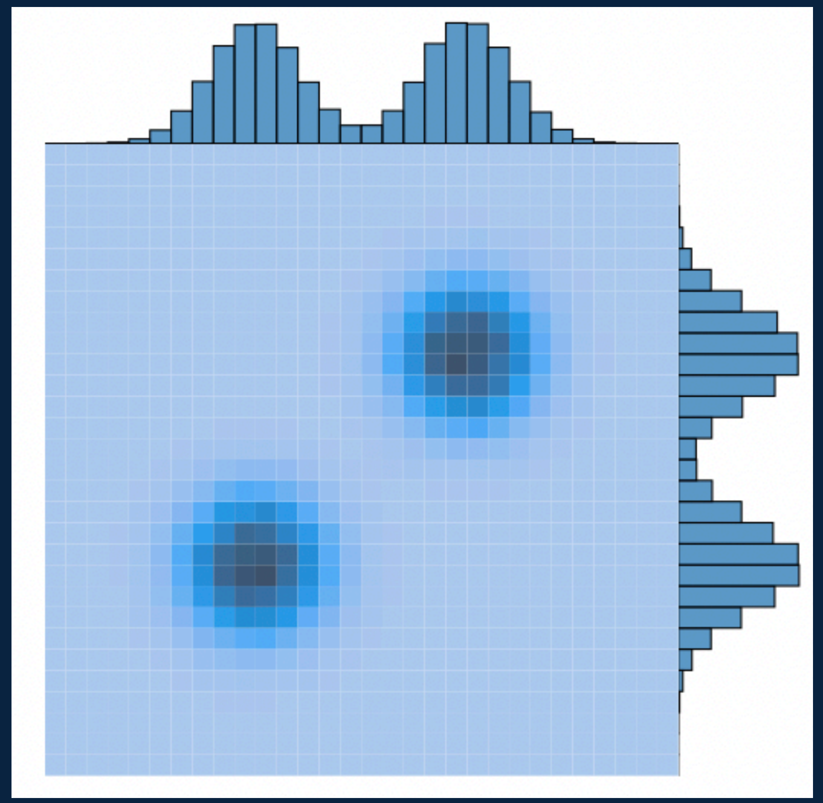
Fit bins with equal probability
and varying width

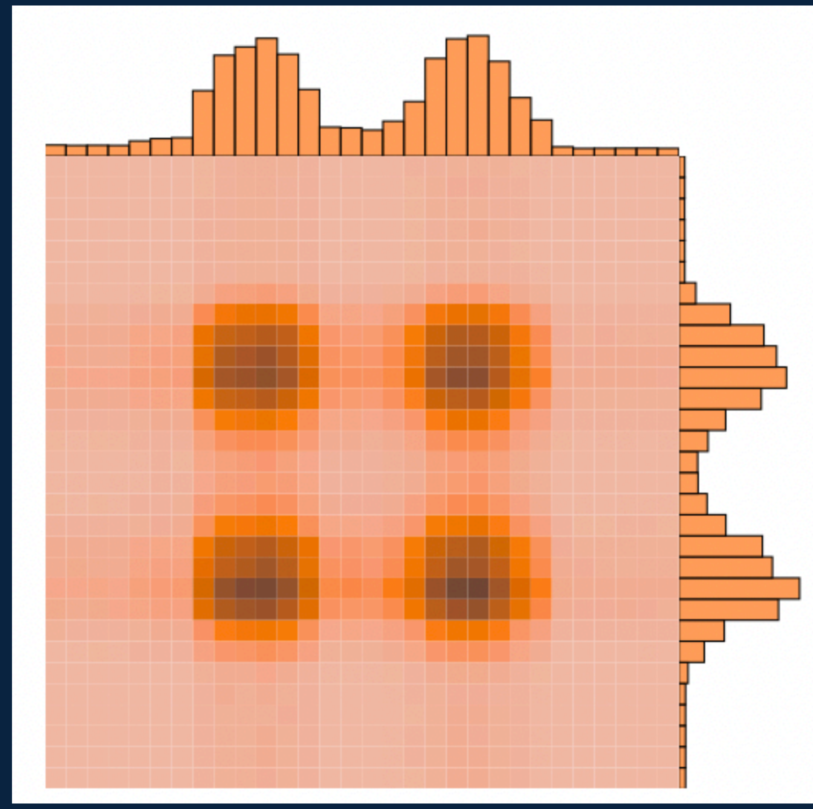


⊕ Computationally cheap

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

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





[G. P. Lepage, 1978]