MadNIS — MadGraph Neural Importance Sampling

Theo Heimel
May 2023

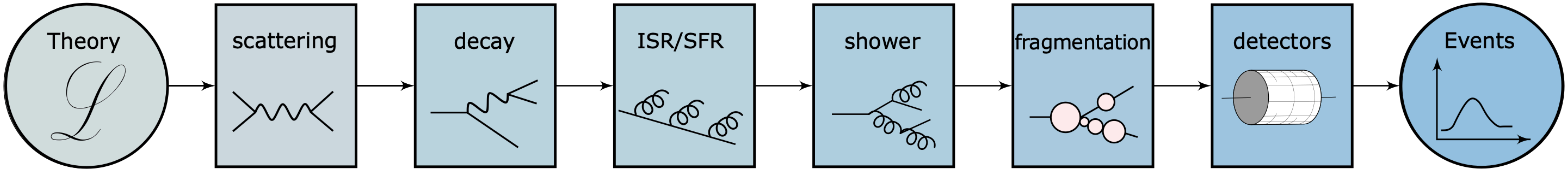
Institut für theoretische Physik
Universität Heidelberg

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

[2212.06172] TH, Winterhalder, Butter, Isaacson, Krause, Maltoni, Mattelaer, Plehn
[23xx.xxxxx] TH, Winterhalder, Maltoni, Mattelaer, Plehn

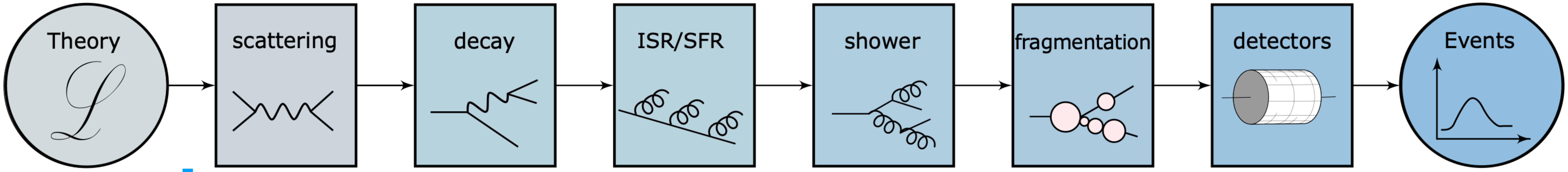
Introduction

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

↓

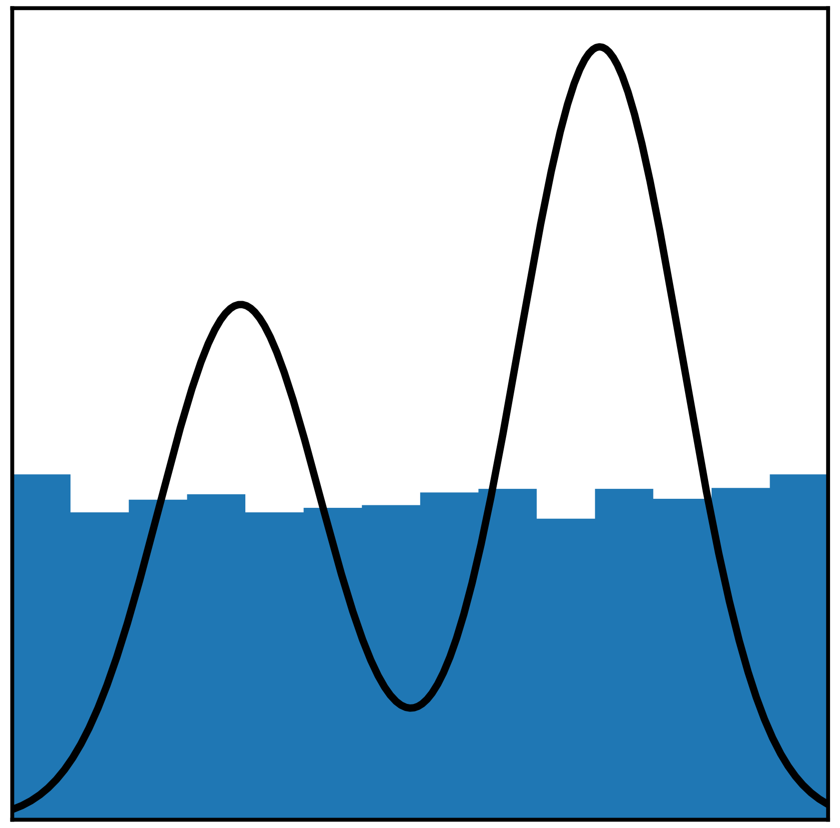
better model
=
faster sampling

Monte Carlo Integration

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

Monte Carlo Integration

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

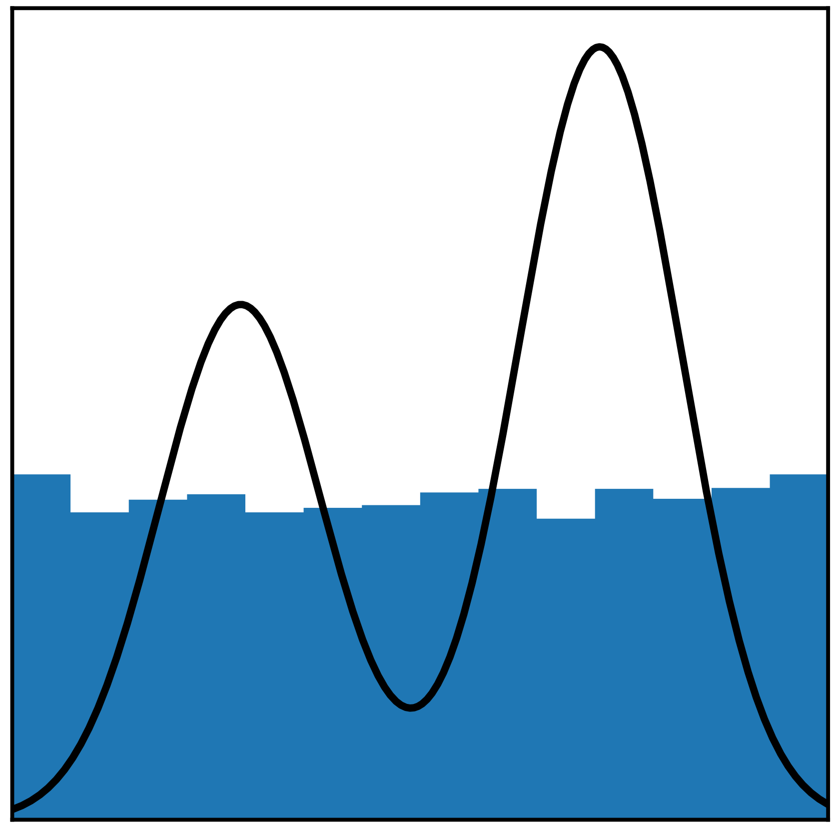


Flat sampling
inefficient

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

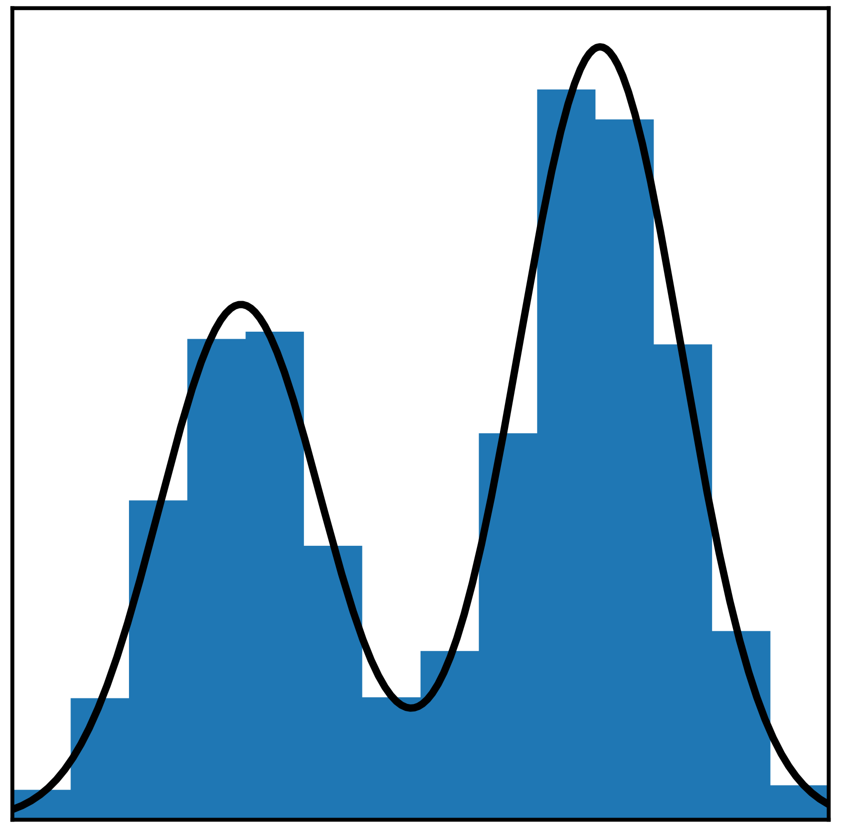
Monte Carlo Integration

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



Flat sampling
inefficient

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

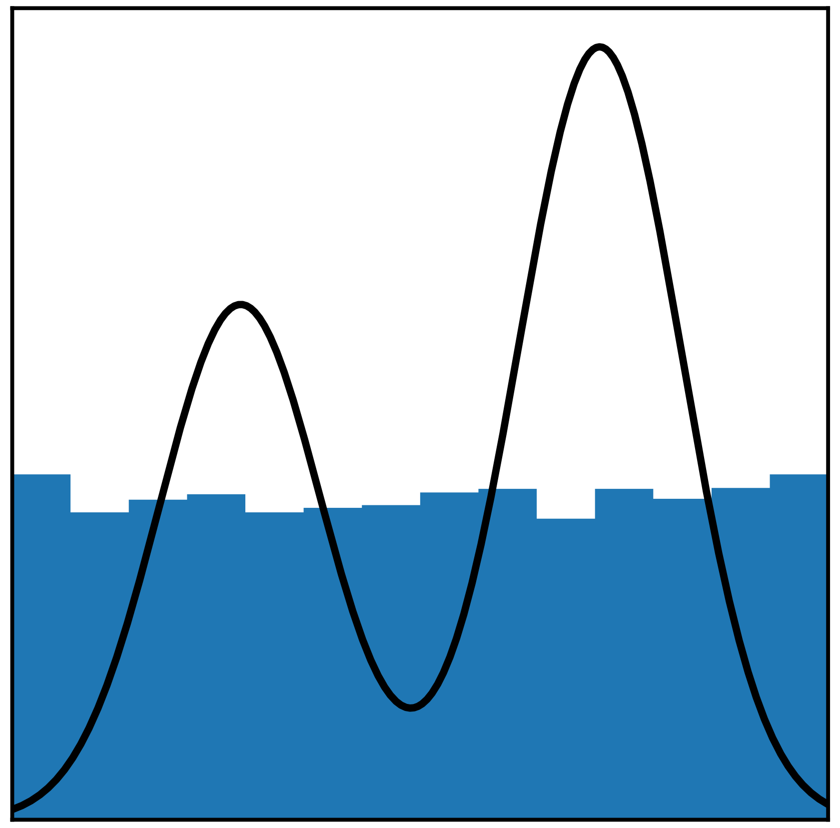


Importance sampling
Find mapping close to integrand

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

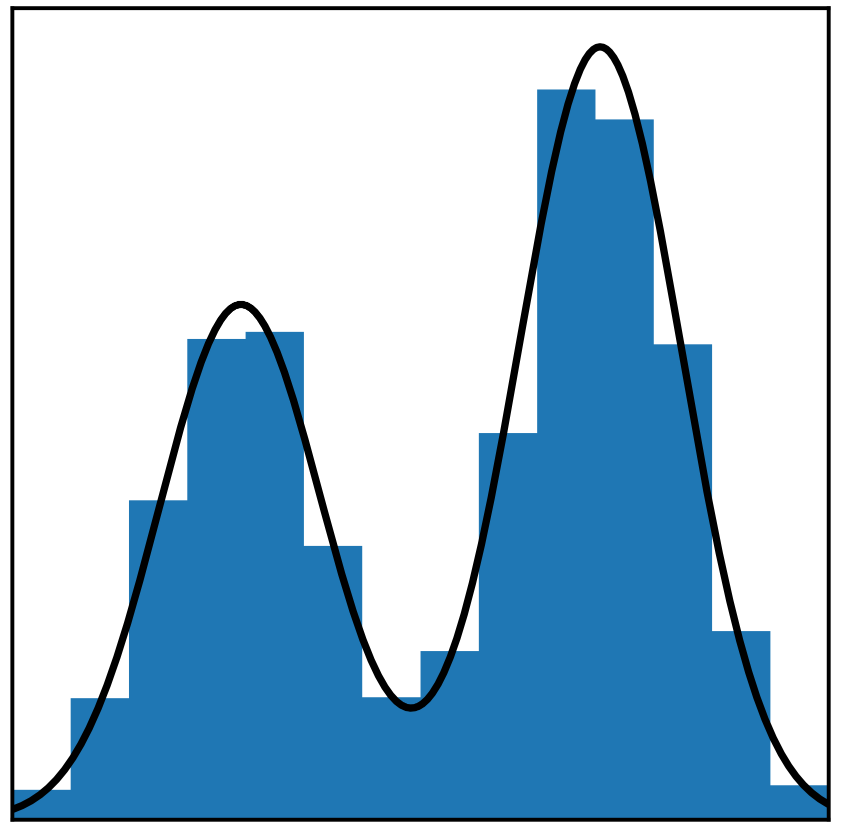
Monte Carlo Integration

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



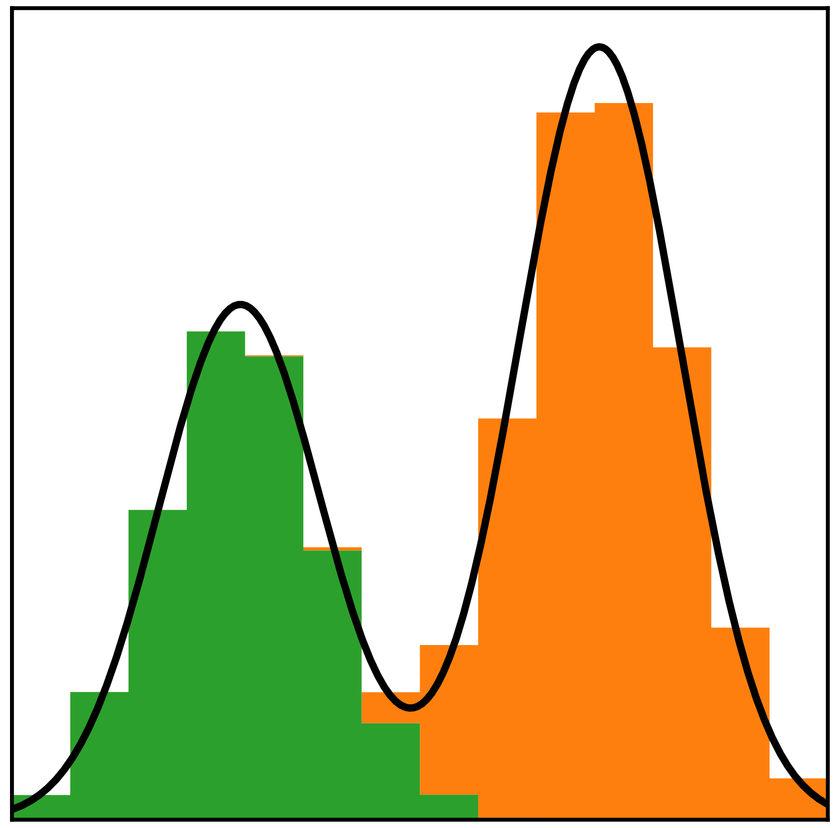
Flat sampling
inefficient

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



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

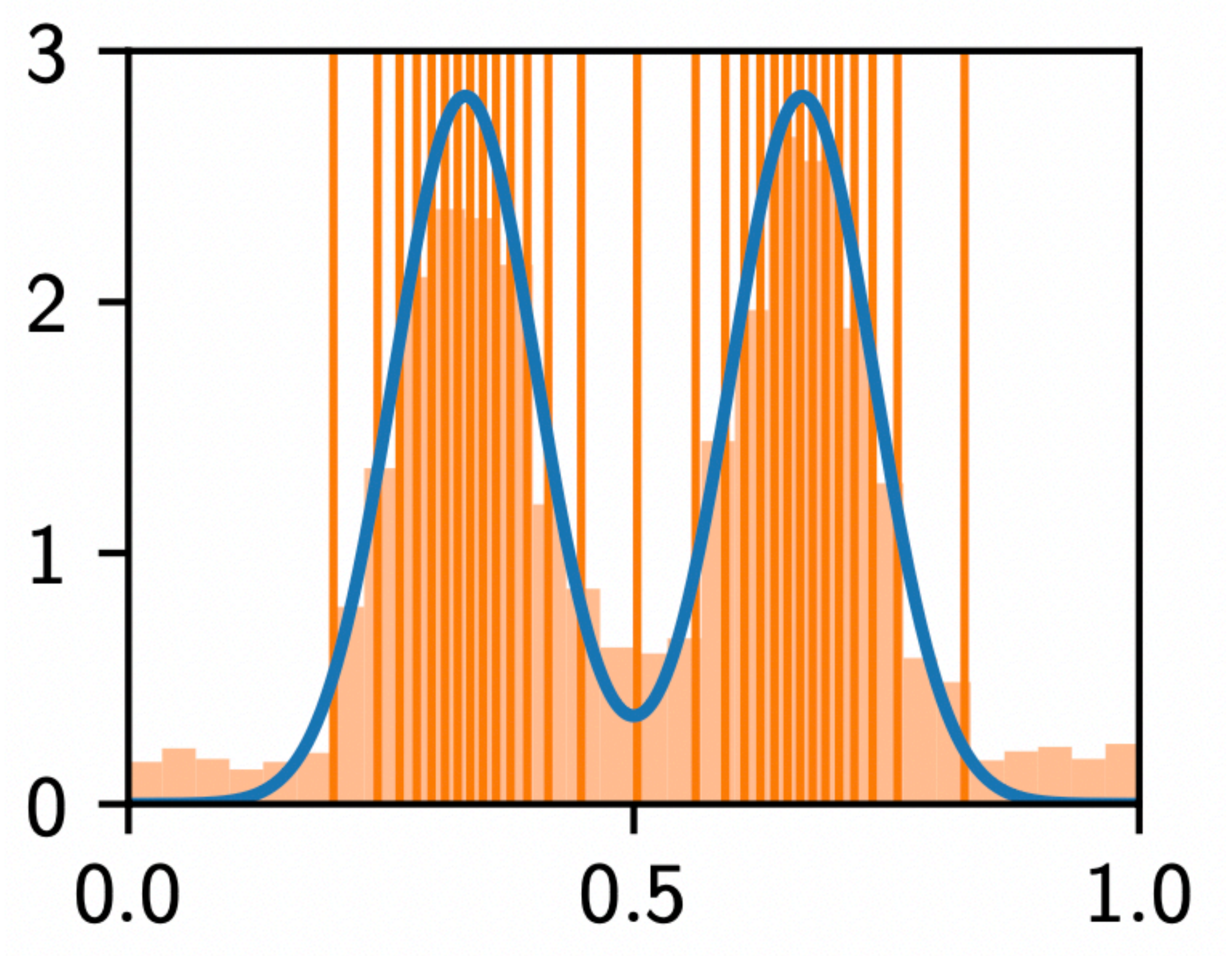
$$I = \sum_i \left\langle \alpha_i(x) \frac{f(x)}{g_i(x)} \right\rangle_{x \sim g_i(x)}$$

VEGAS algorithm

Factorize probability

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

Fit bins with equal probability
and varying width

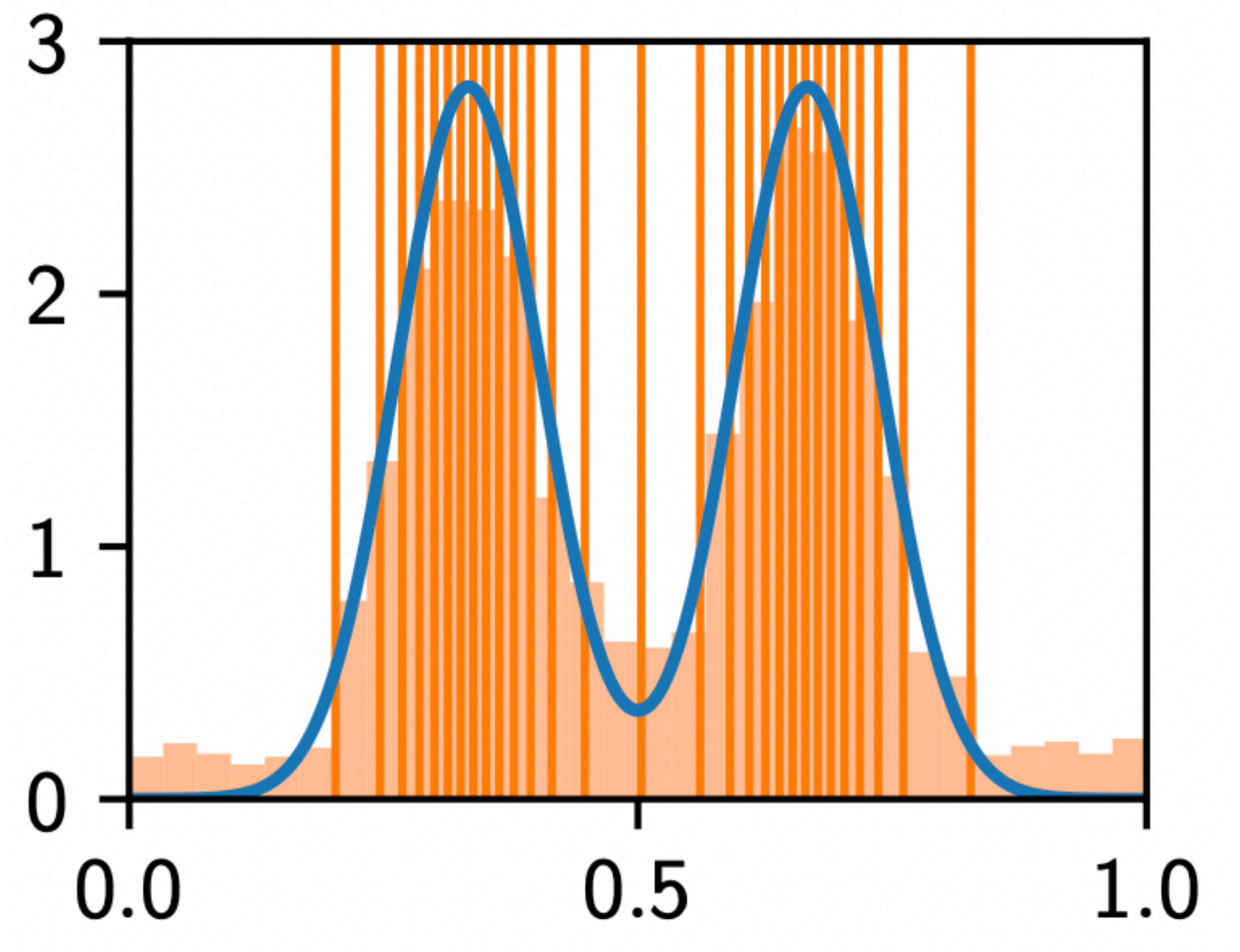


[G. P. Lepage, 1978]

VEGAS algorithm

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

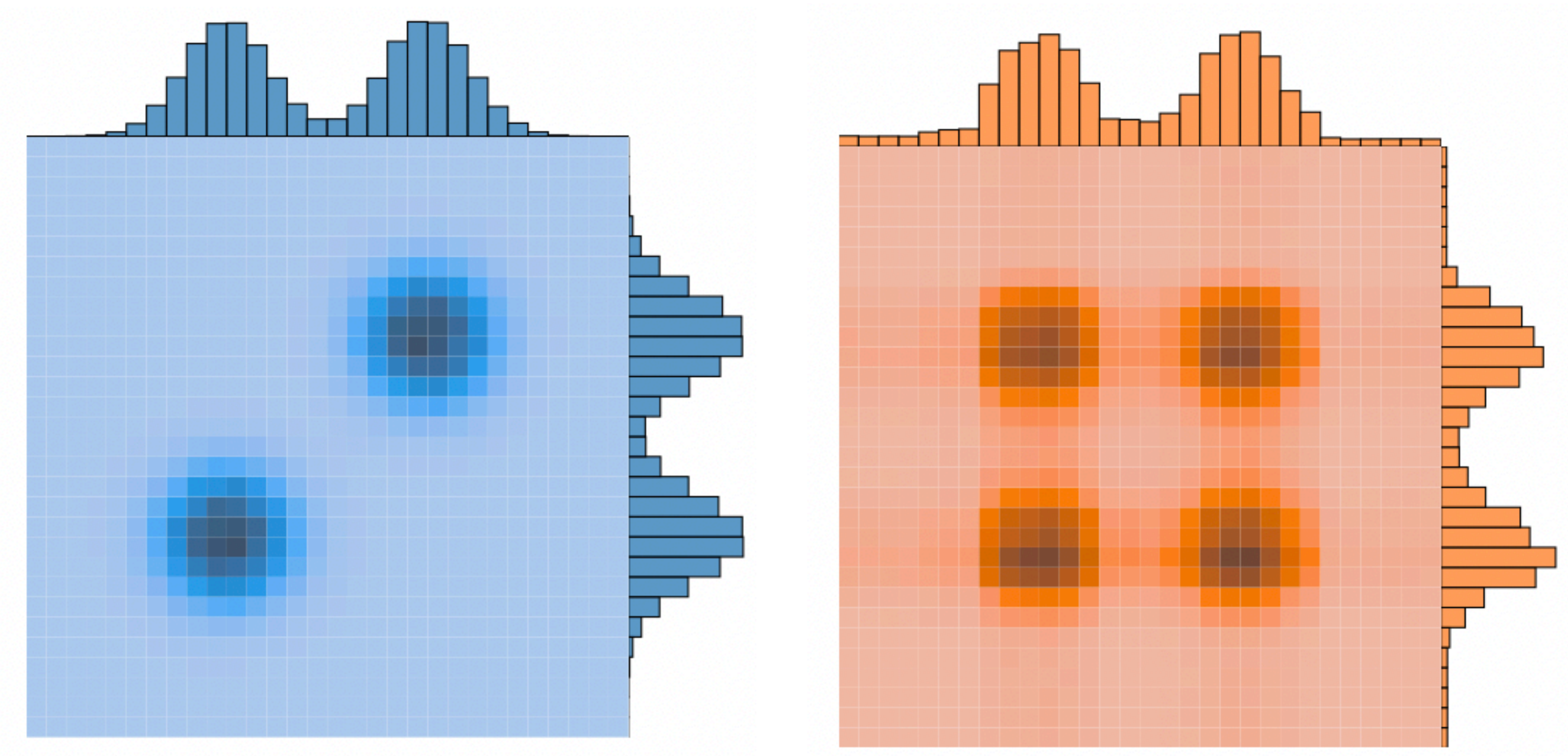
Fit bins with equal probability
and varying width



⊕ Computationally cheap

⊖ High-dim and rich peaking functions
→ slow convergence

⊖ Peaks not aligned with grid axes
→ phantom peaks

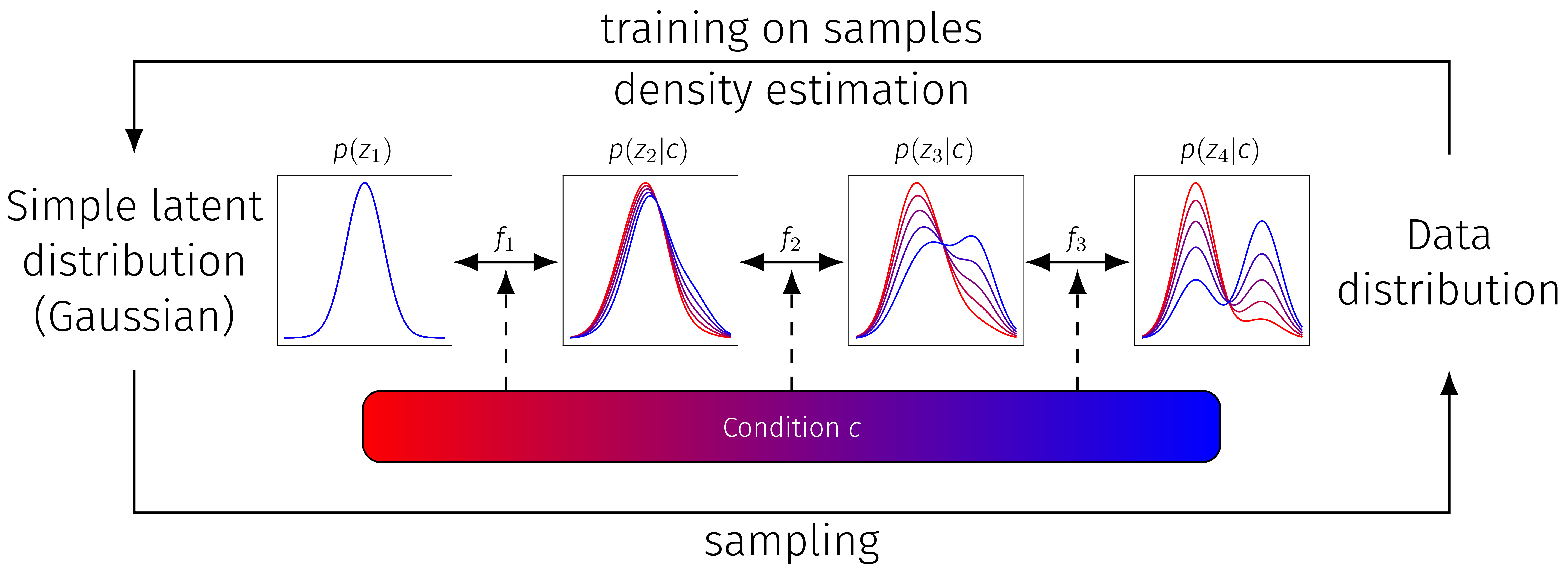


[G. P. Lepage, 1978]

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]

MadNIS: Neural Importance Sampling

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

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

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

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

MadNIS: Overview

MADNIS

Basic functionality

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

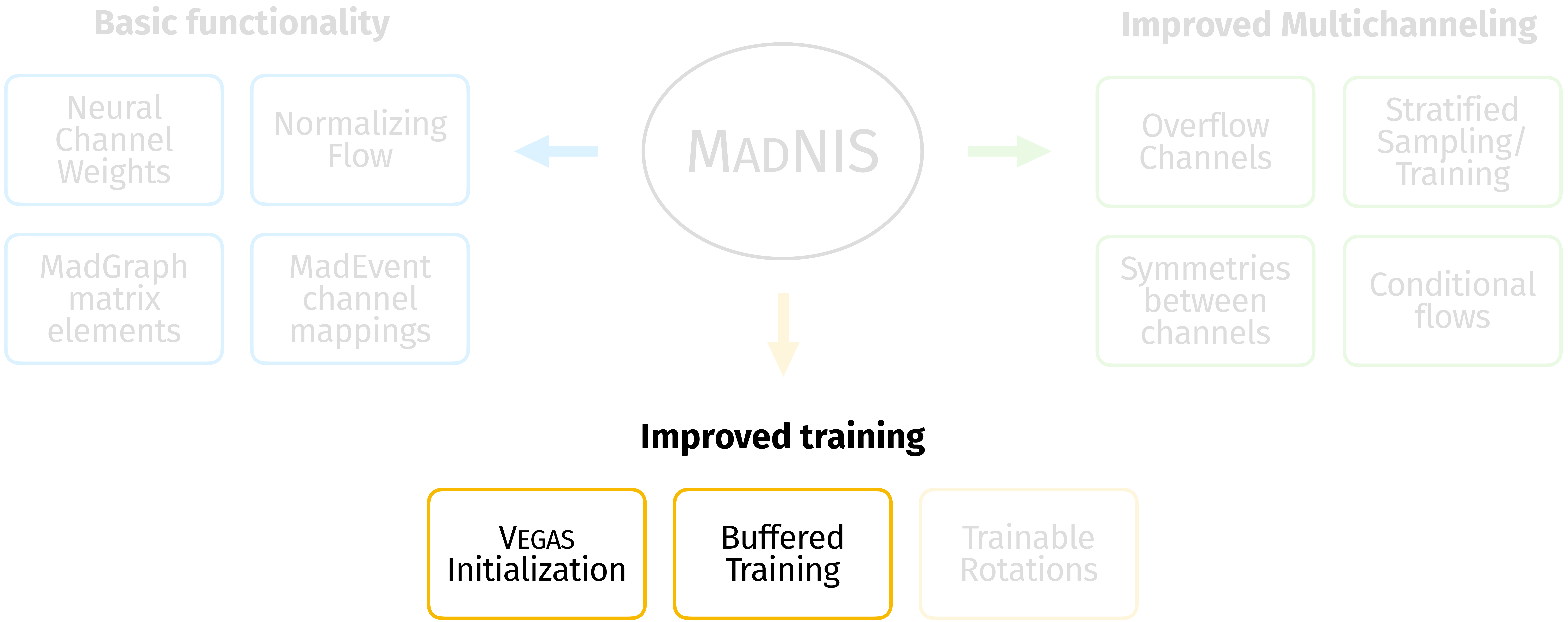
Improved Multichanneling

- Overflow Channels
- Stratified Sampling/ Training
- Symmetries between channels
- Conditional flows

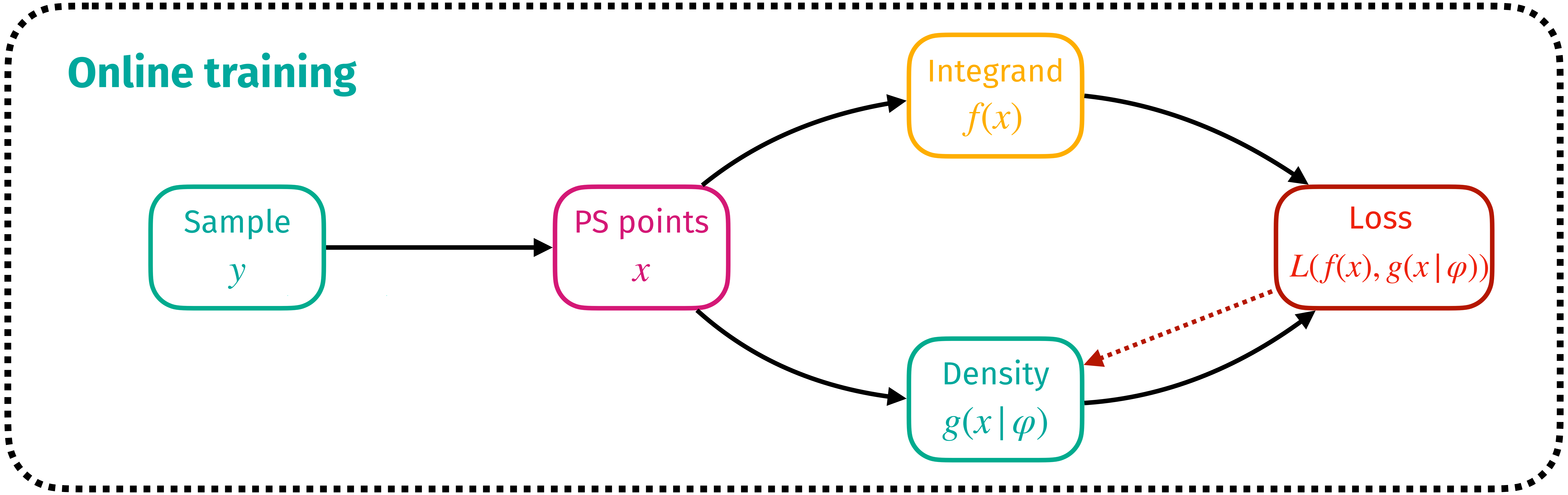
Improved training

- VEGAS Initialization
- Buffered Training
- Trainable Rotations

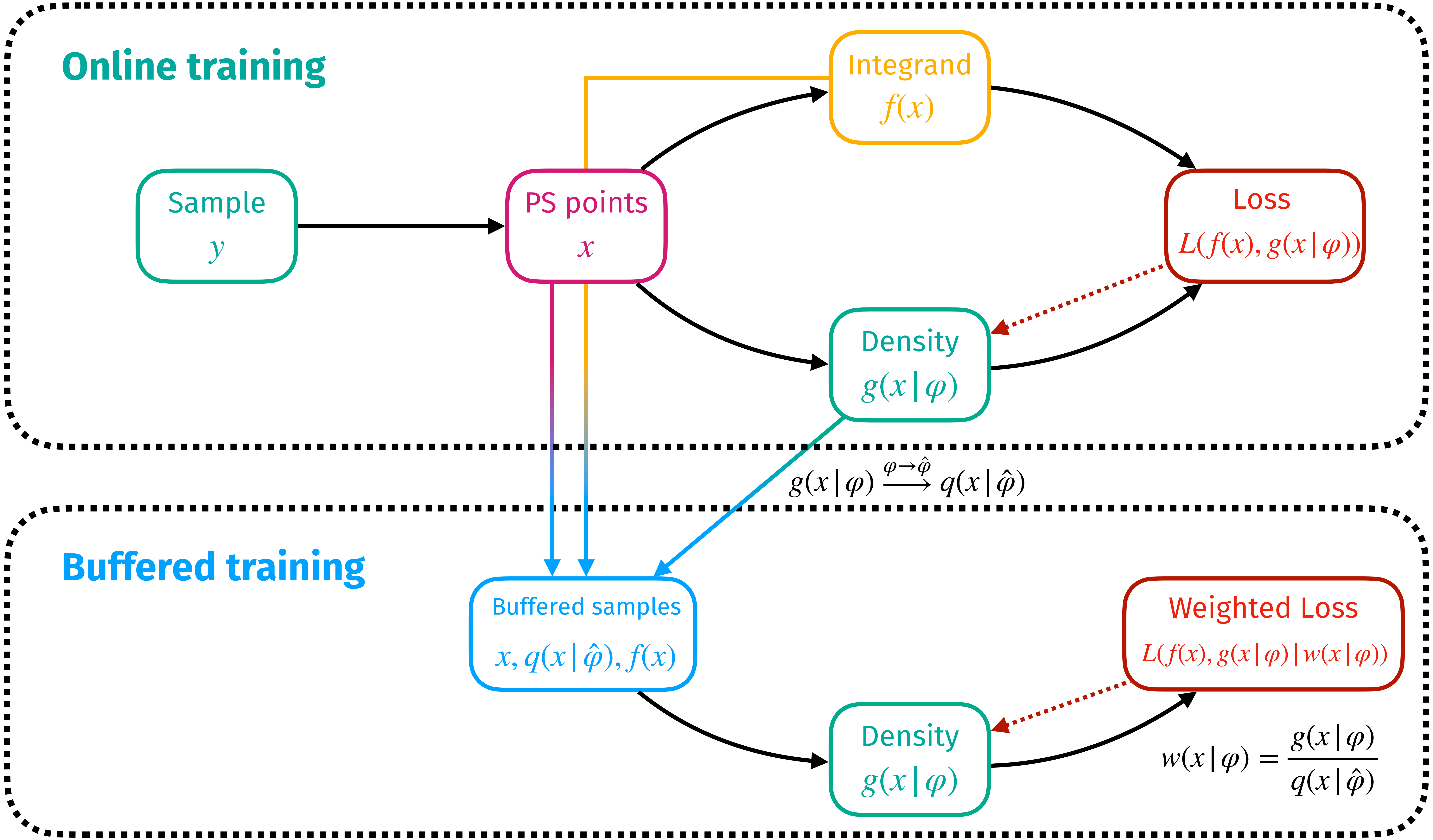
MadNIS: Overview



Buffered Training



Buffered Training



Buffered Training

Training algorithm

generate new samples, train on them, save samples

↓

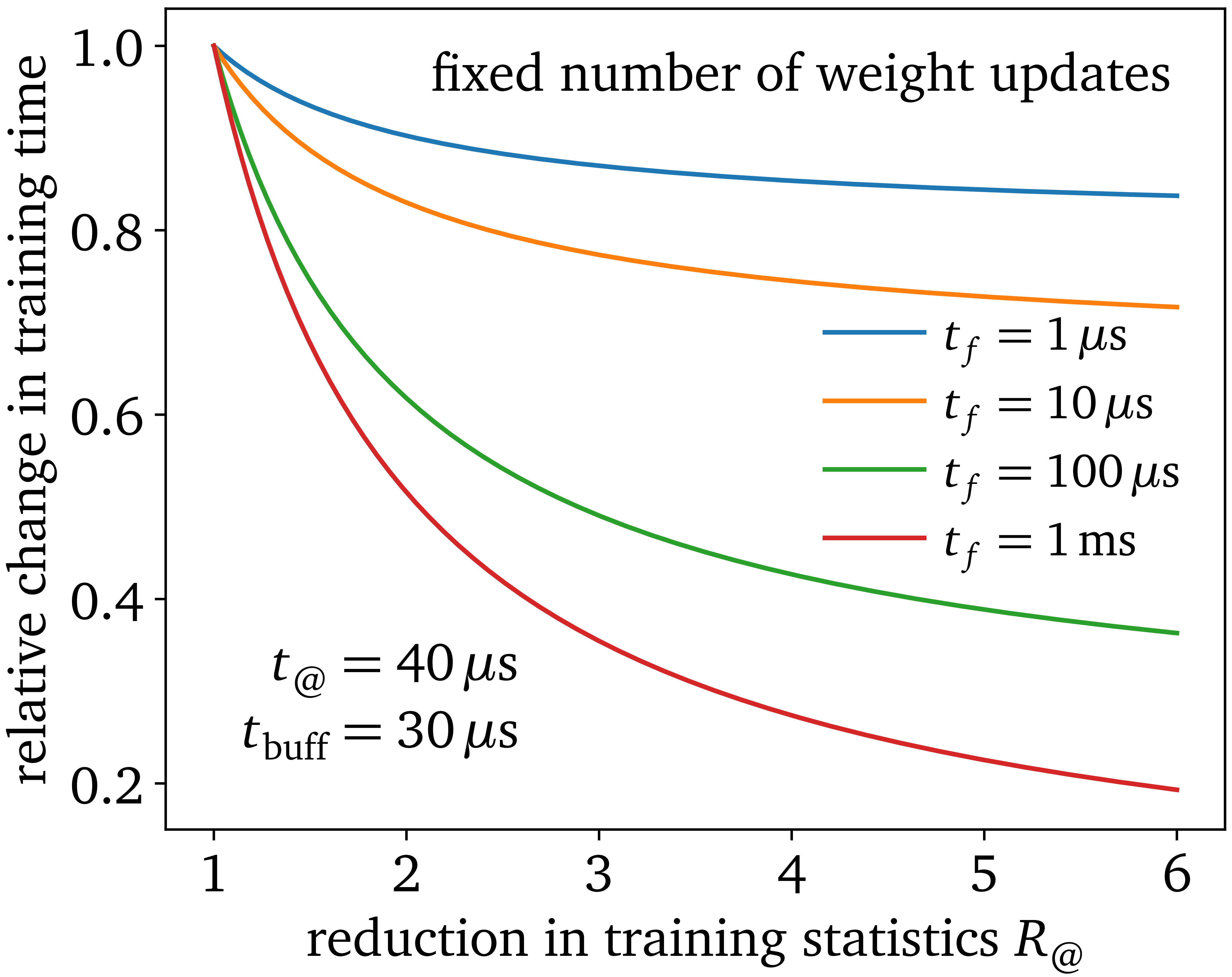
train on saved samples n times

↓

repeat

Reduction in training statistics by

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



VEGAS Initialization

	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

↓

Combine advantages:

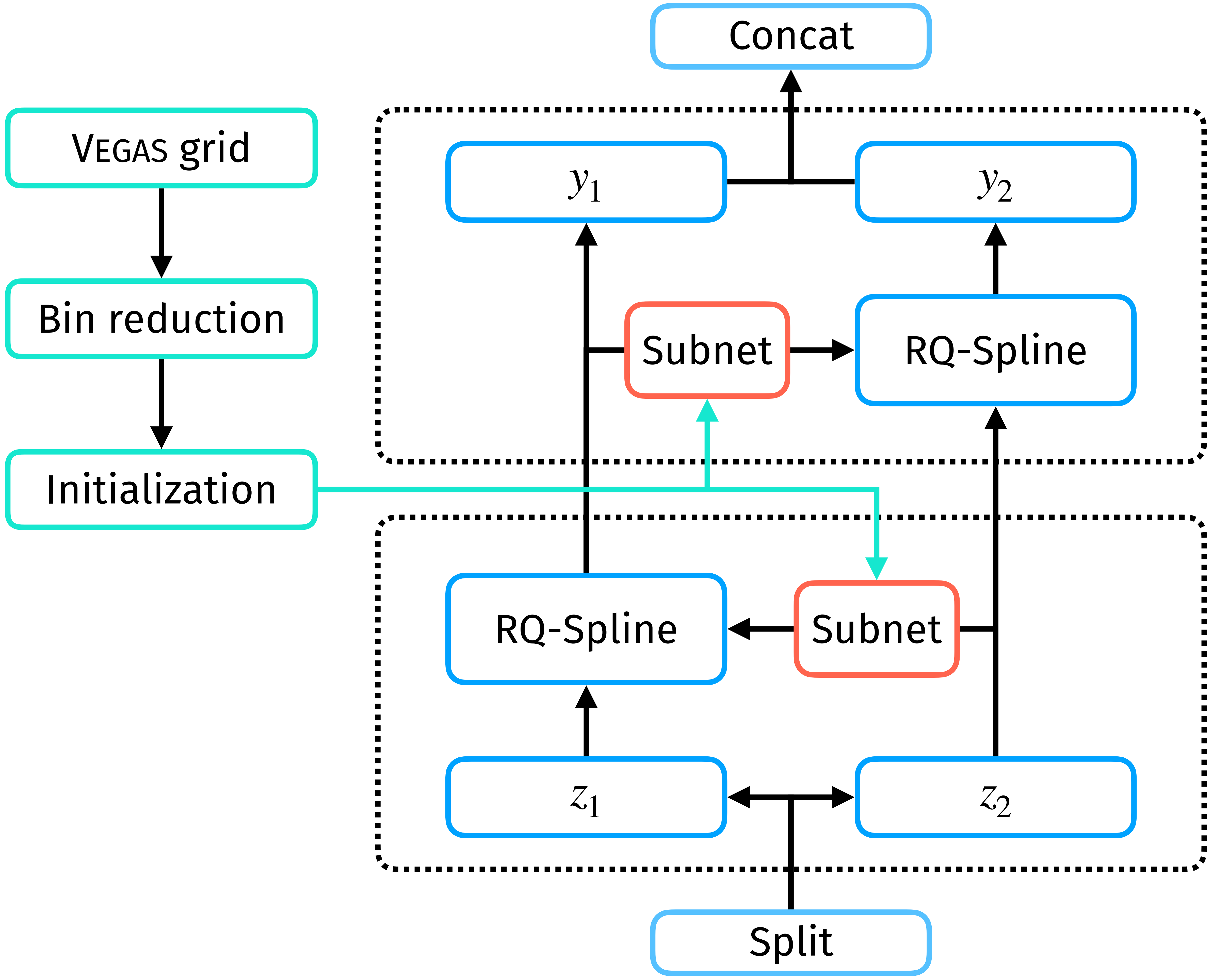
Pre-trained VEGAS grid as starting point for flow training

VEGAS Initialization

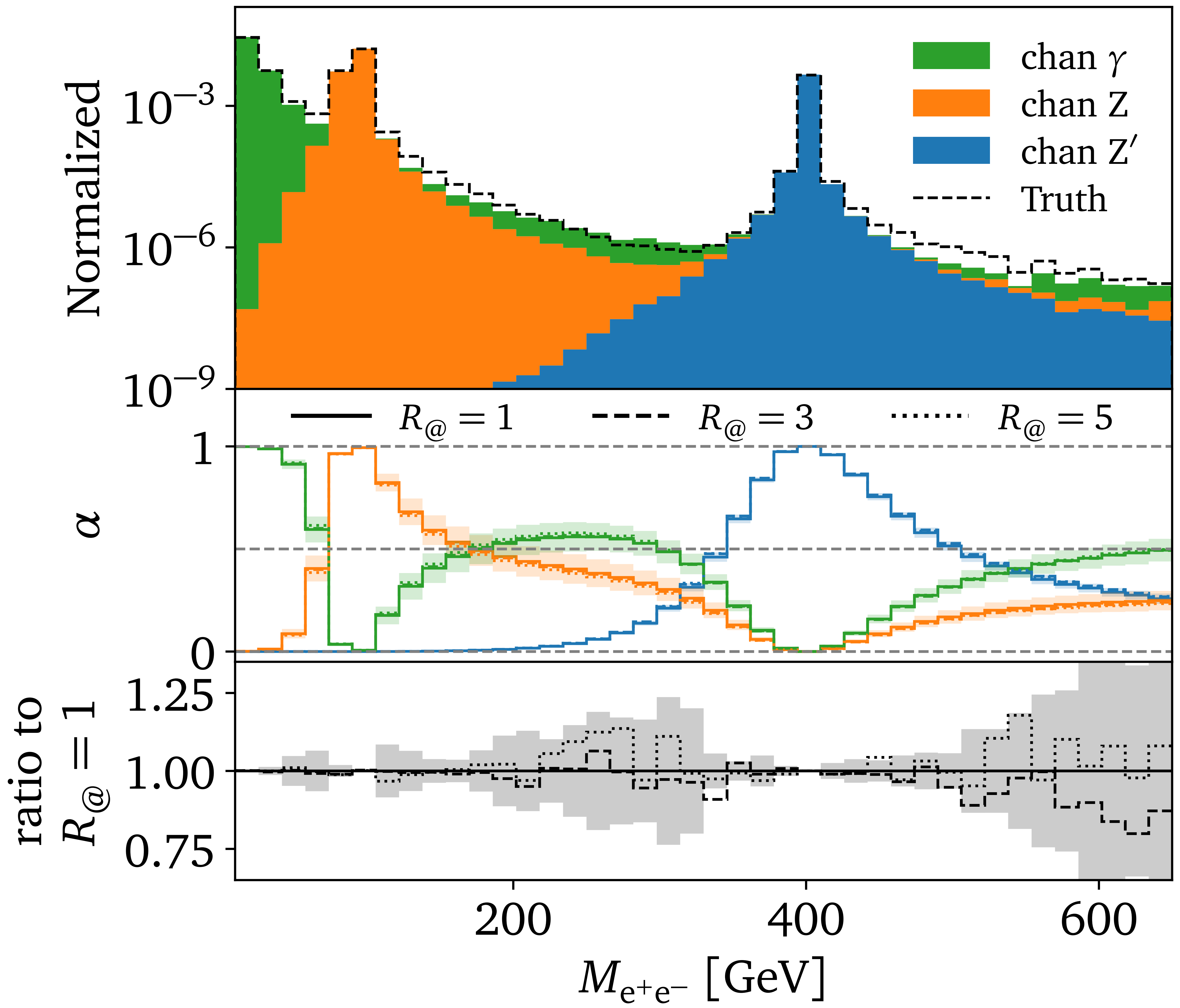
	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

Combine advantages:

Pre-trained VEGAS grid as starting point for flow training

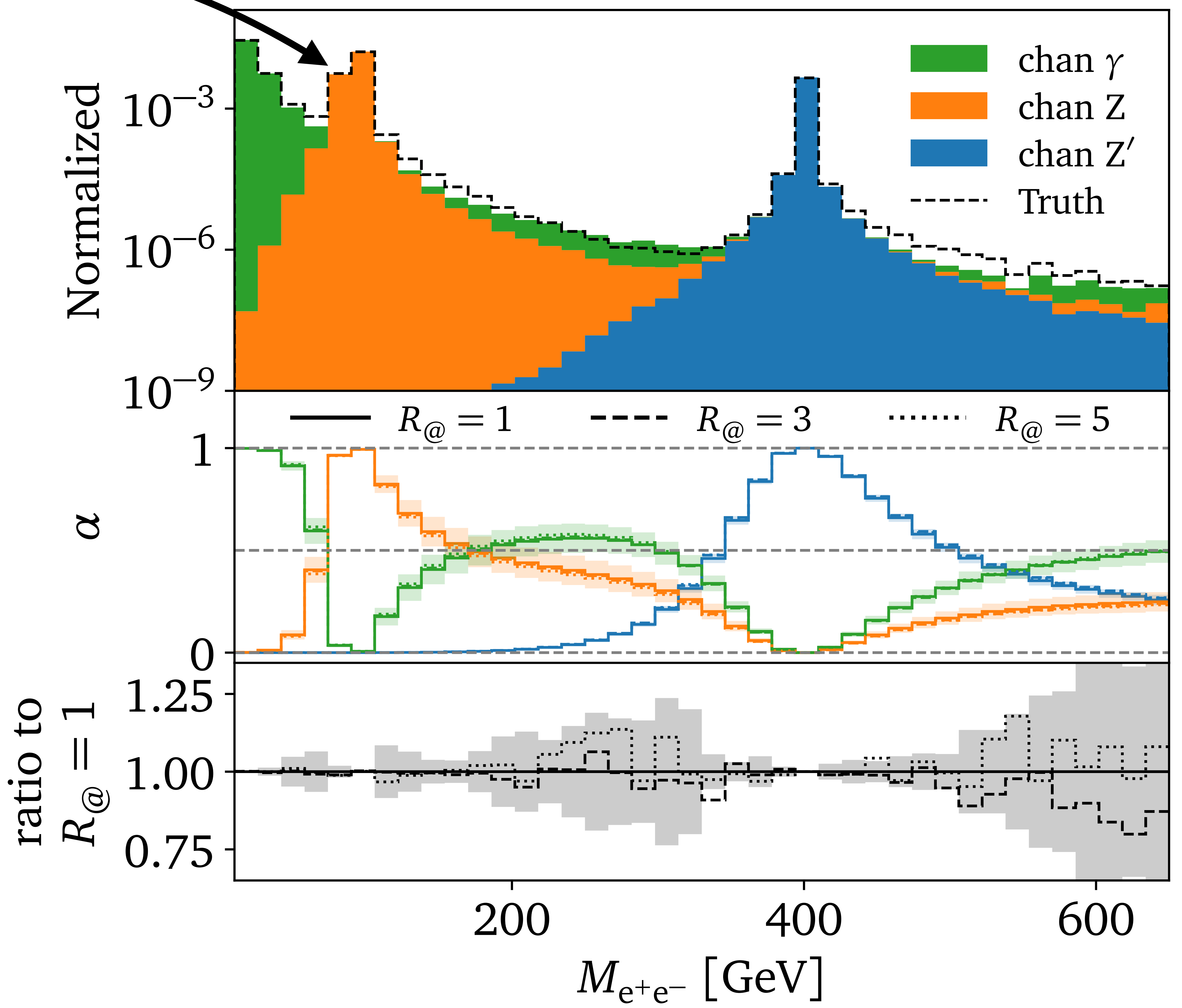


Toy Example: Drell-Yan + Z’



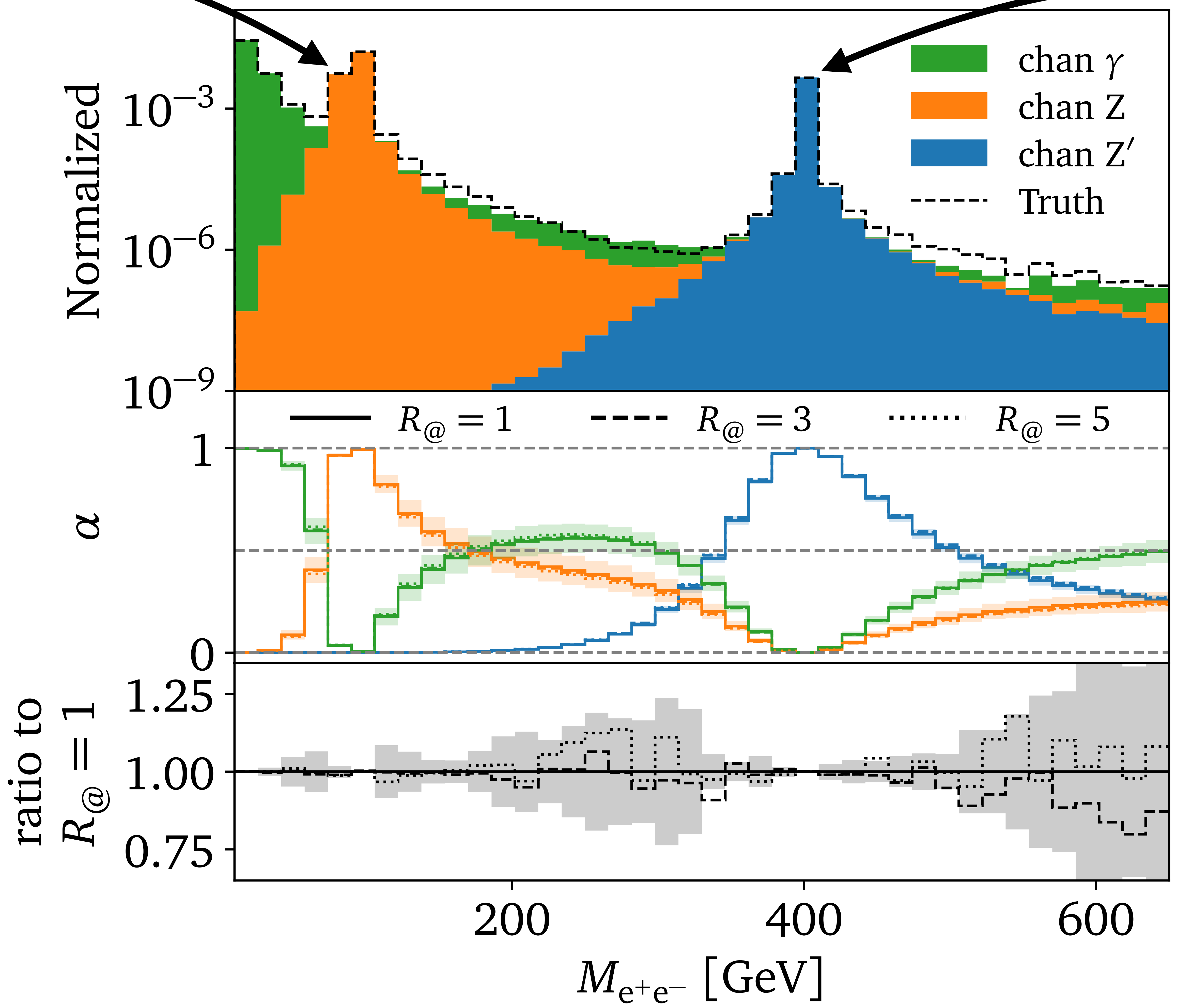
Toy Example: Drell-Yan + Z’

Learned distribution matches truth



Toy Example: Drell-Yan + Z’

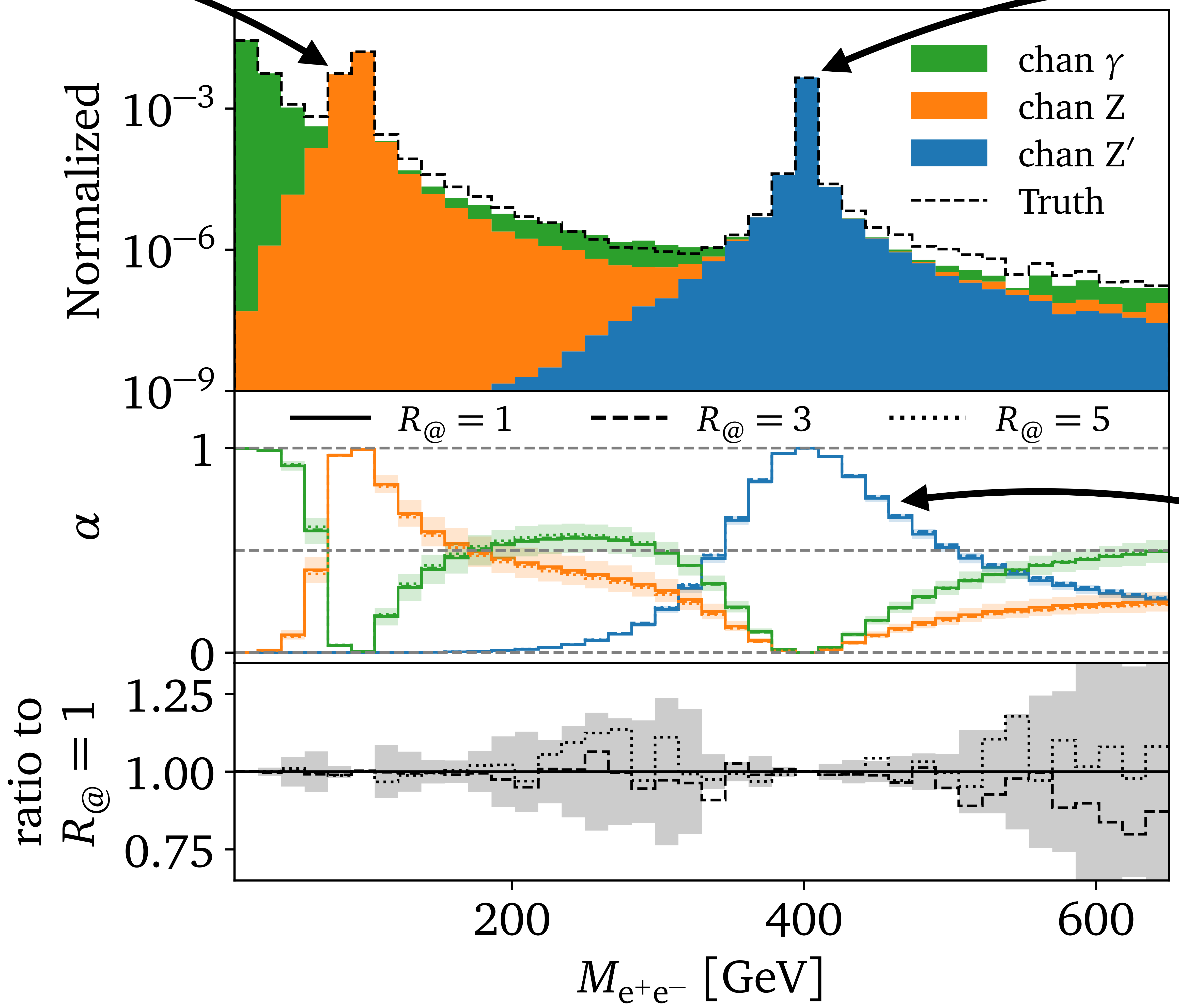
Learned distribution matches truth



Peaks mapped out by different channels

Toy Example: Drell-Yan + Z'

Learned distribution matches truth



Peaks mapped out by different channels

Channel weights learned by network

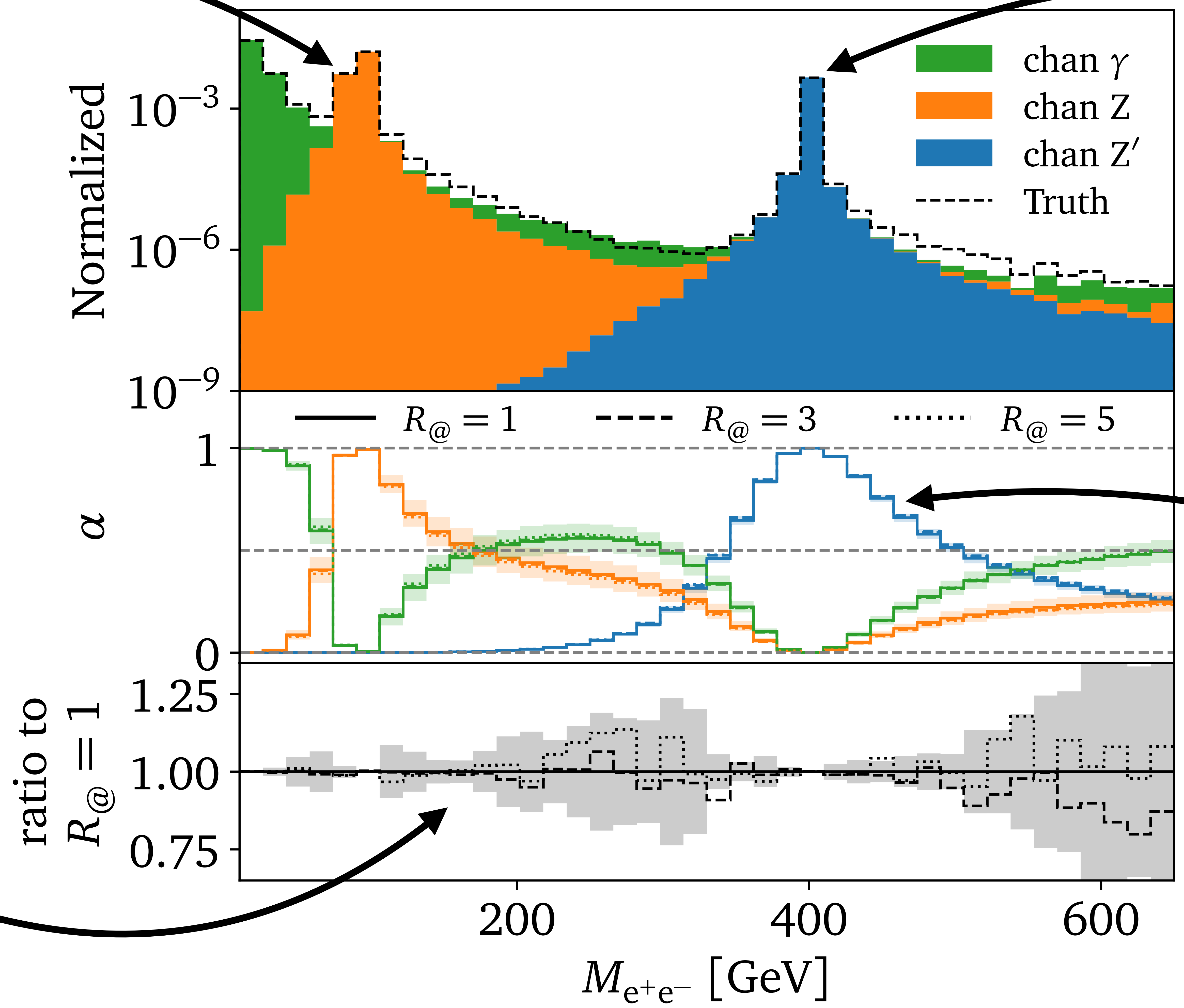
Toy Example: Drell-Yan + Z’

Learned distribution matches truth

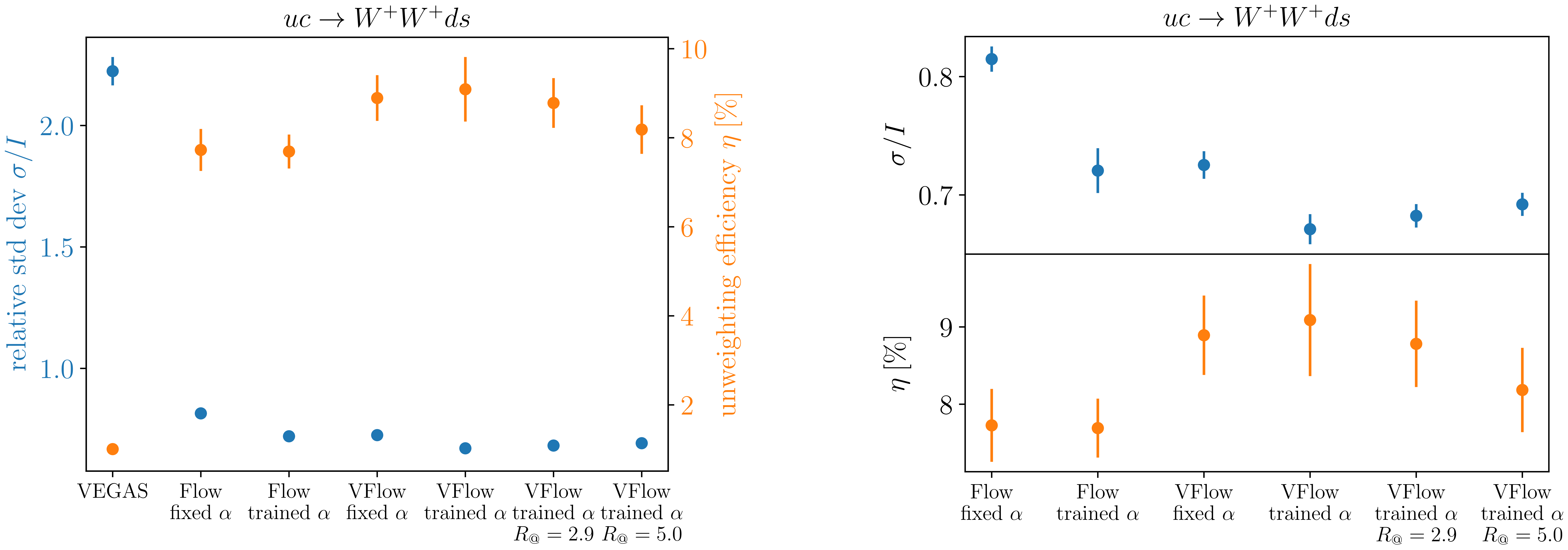
Peaks mapped out by different channels

Channel weights learned by network

Use samples multiple times to make training faster

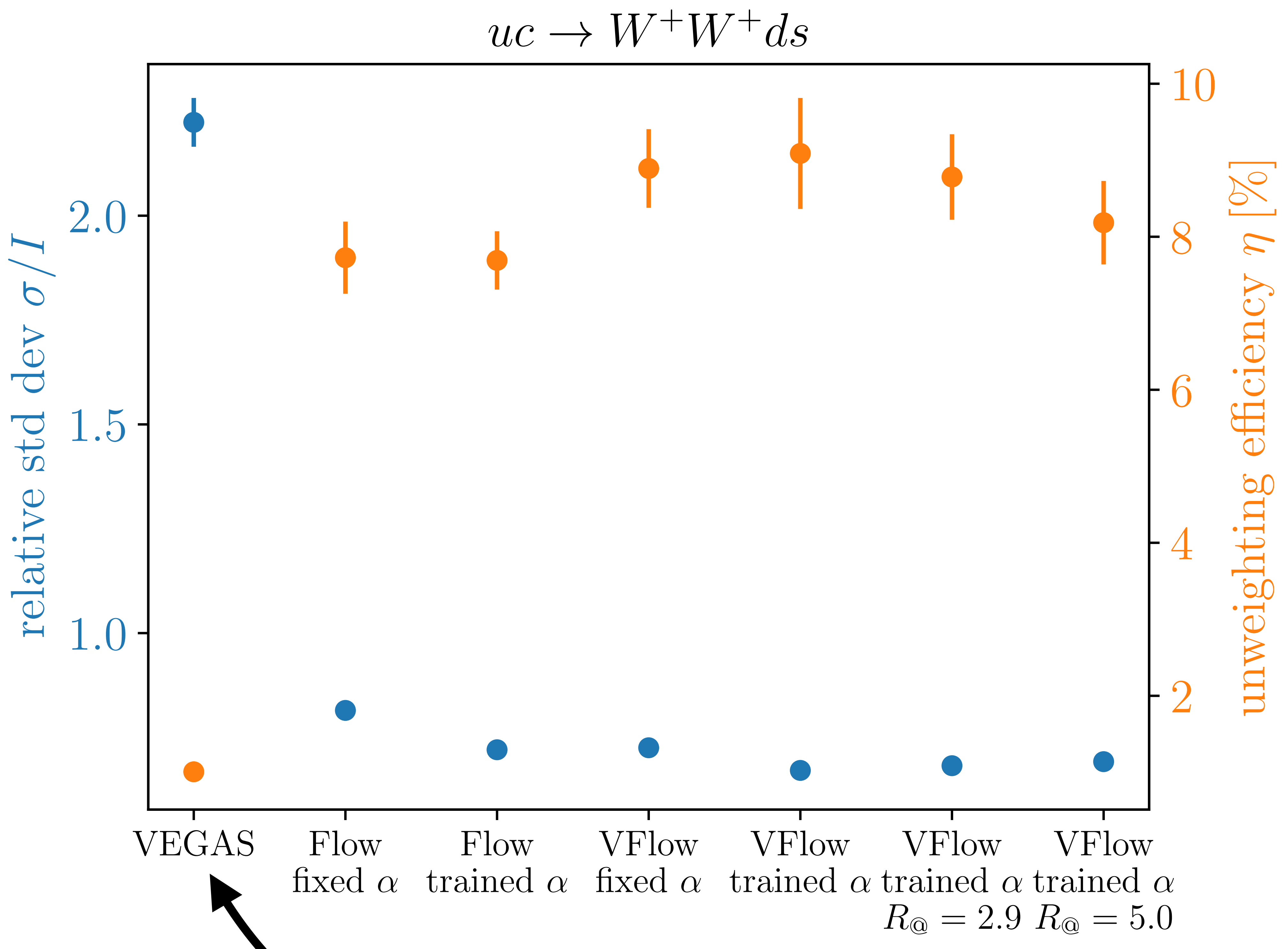


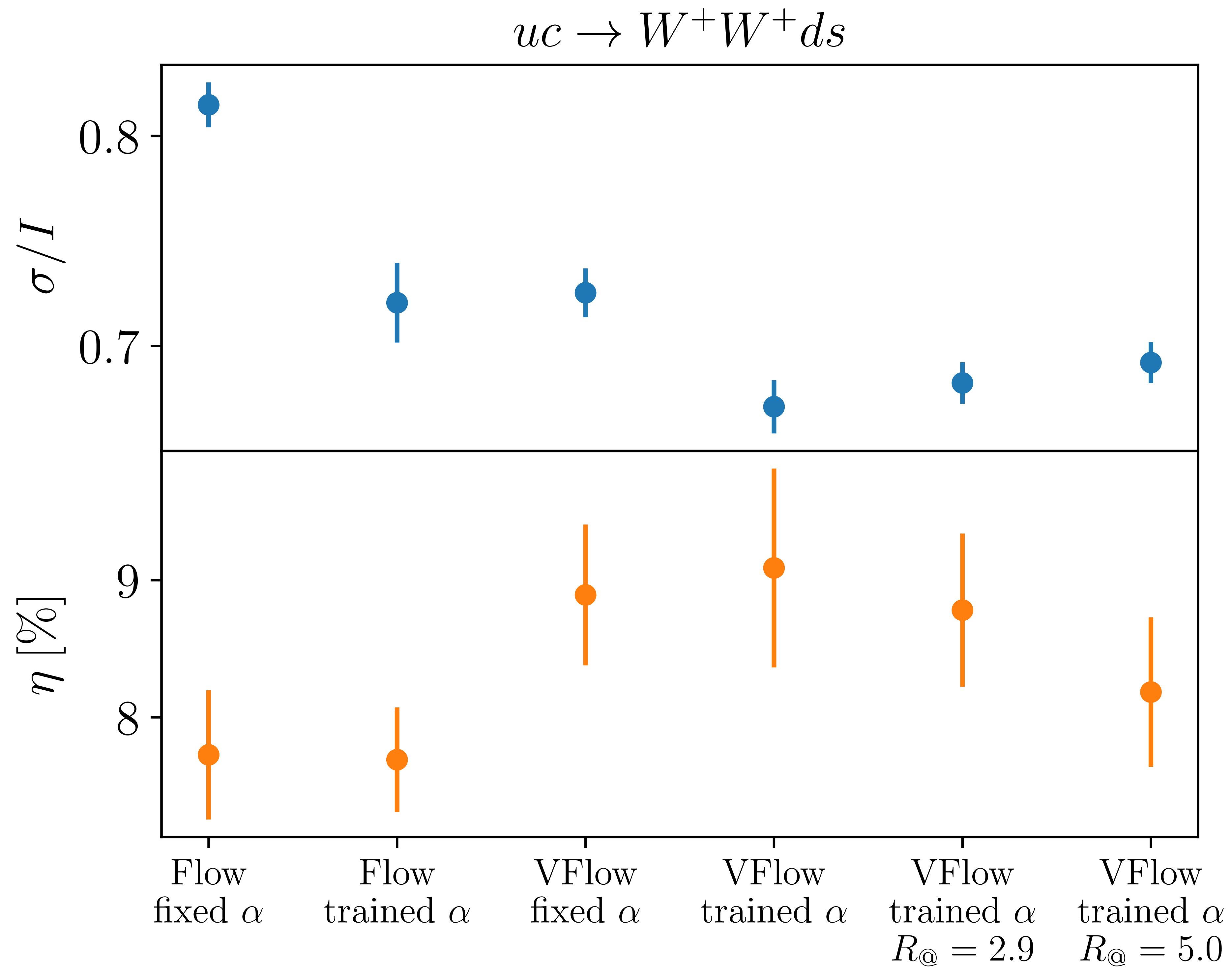
LHC Example: Vector Boson Scattering



(preliminary)

LHC Example: Vector Boson Scattering

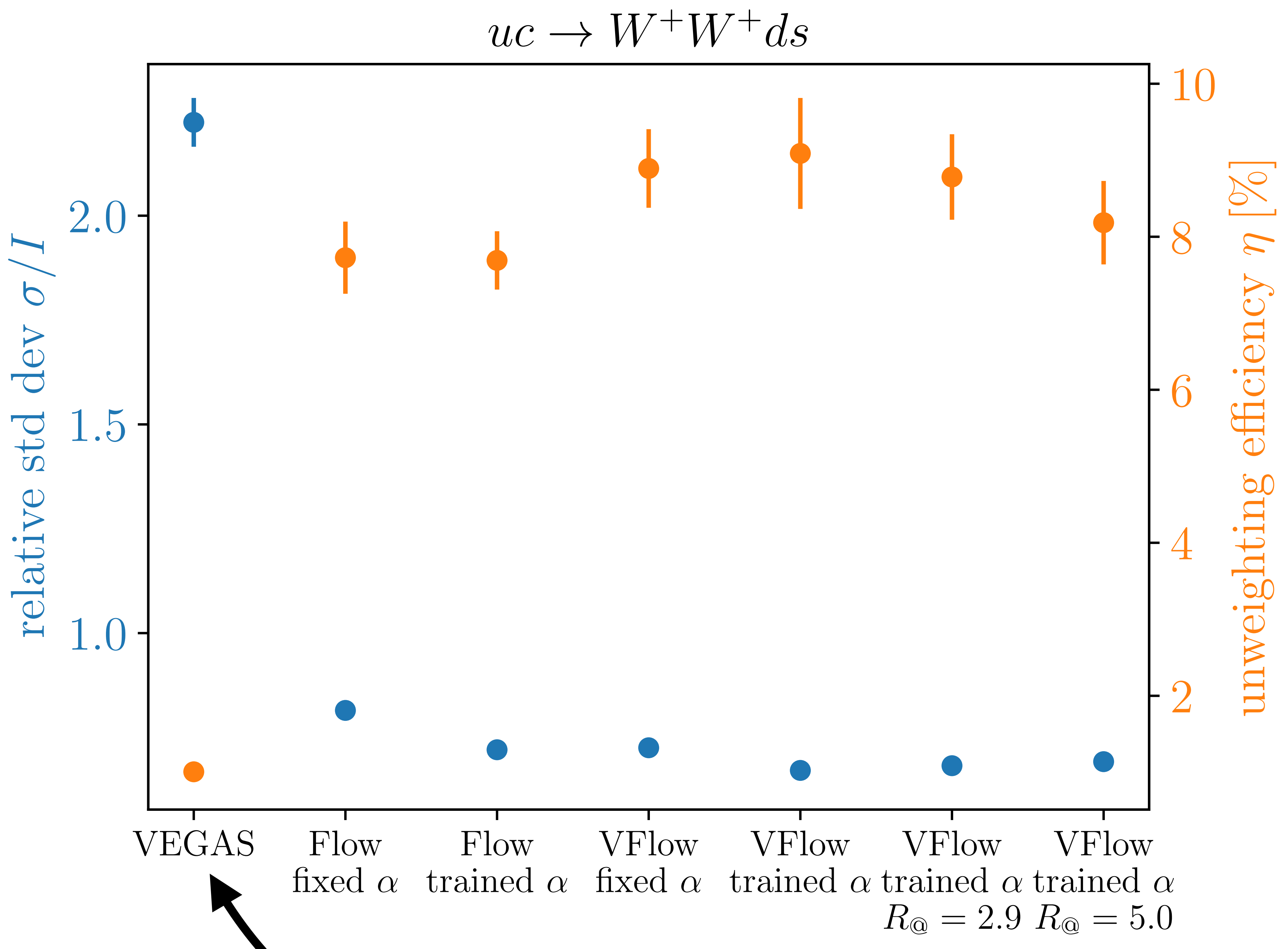




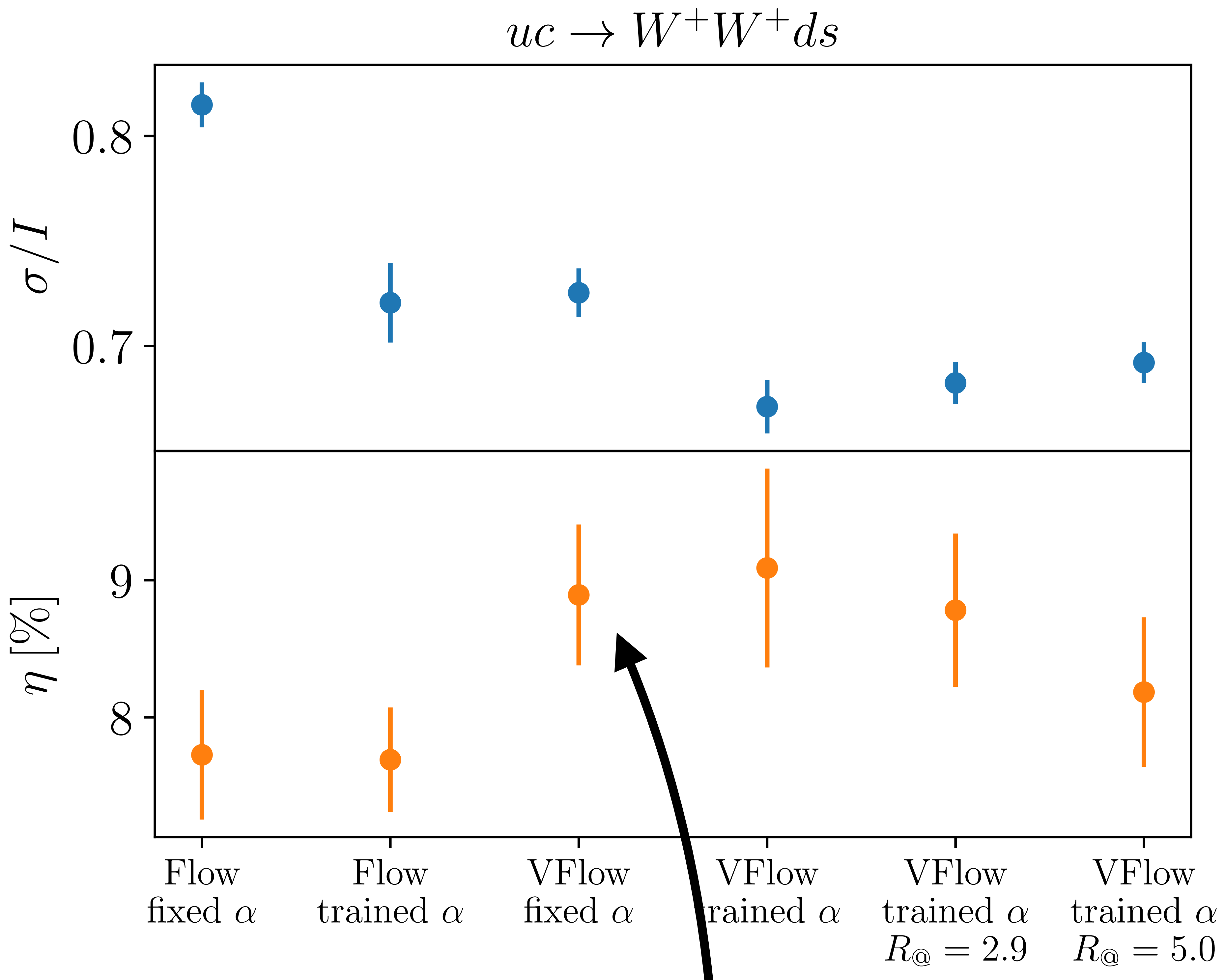
(preliminary)

Unweighting efficiency improved up to factor ~9 compared to VEGAS

LHC Example: Vector Boson Scattering



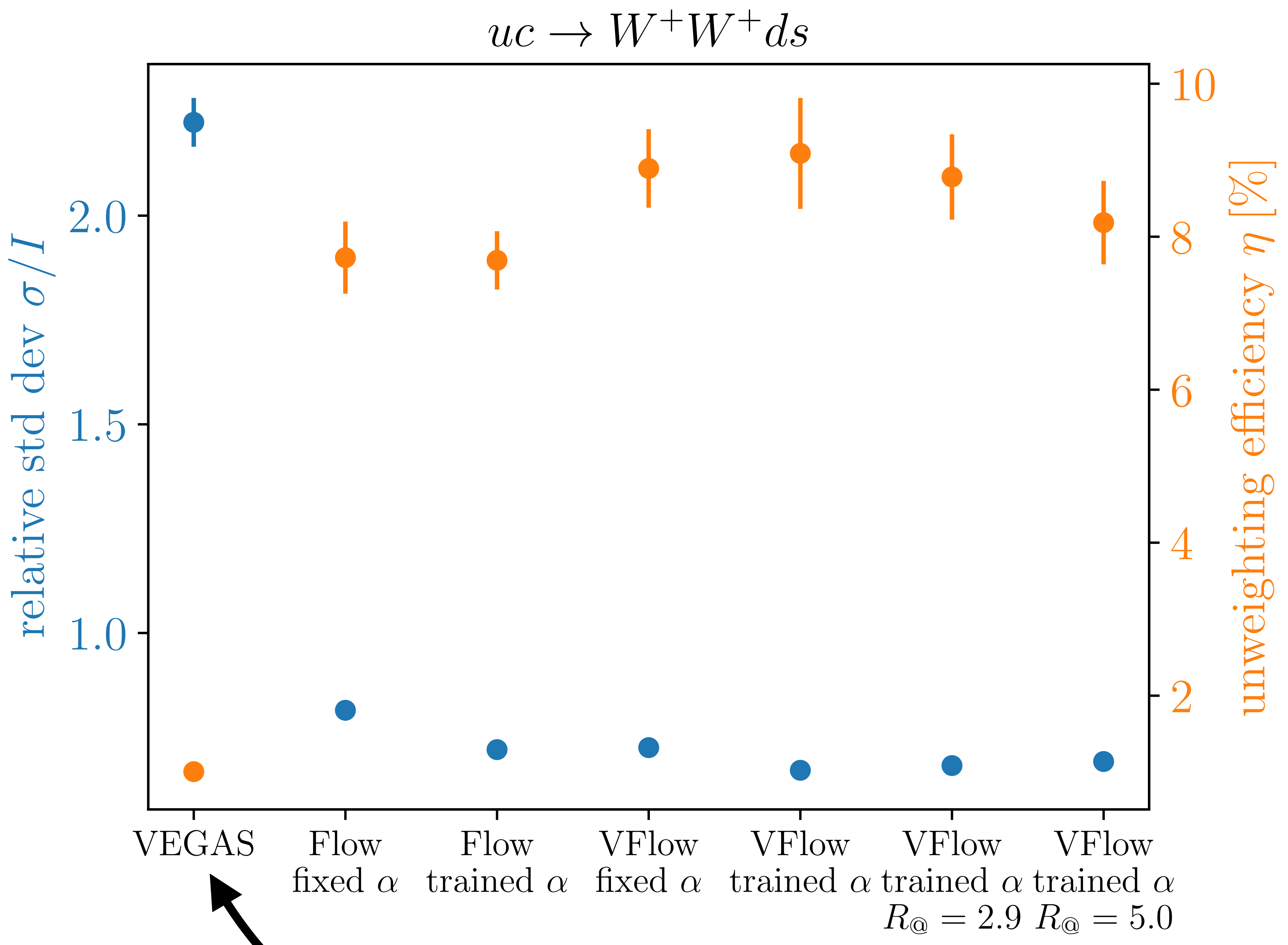
Unweighting efficiency improved up to factor ~9 compared to VEGAS



Big improvement from VEGAS initialization

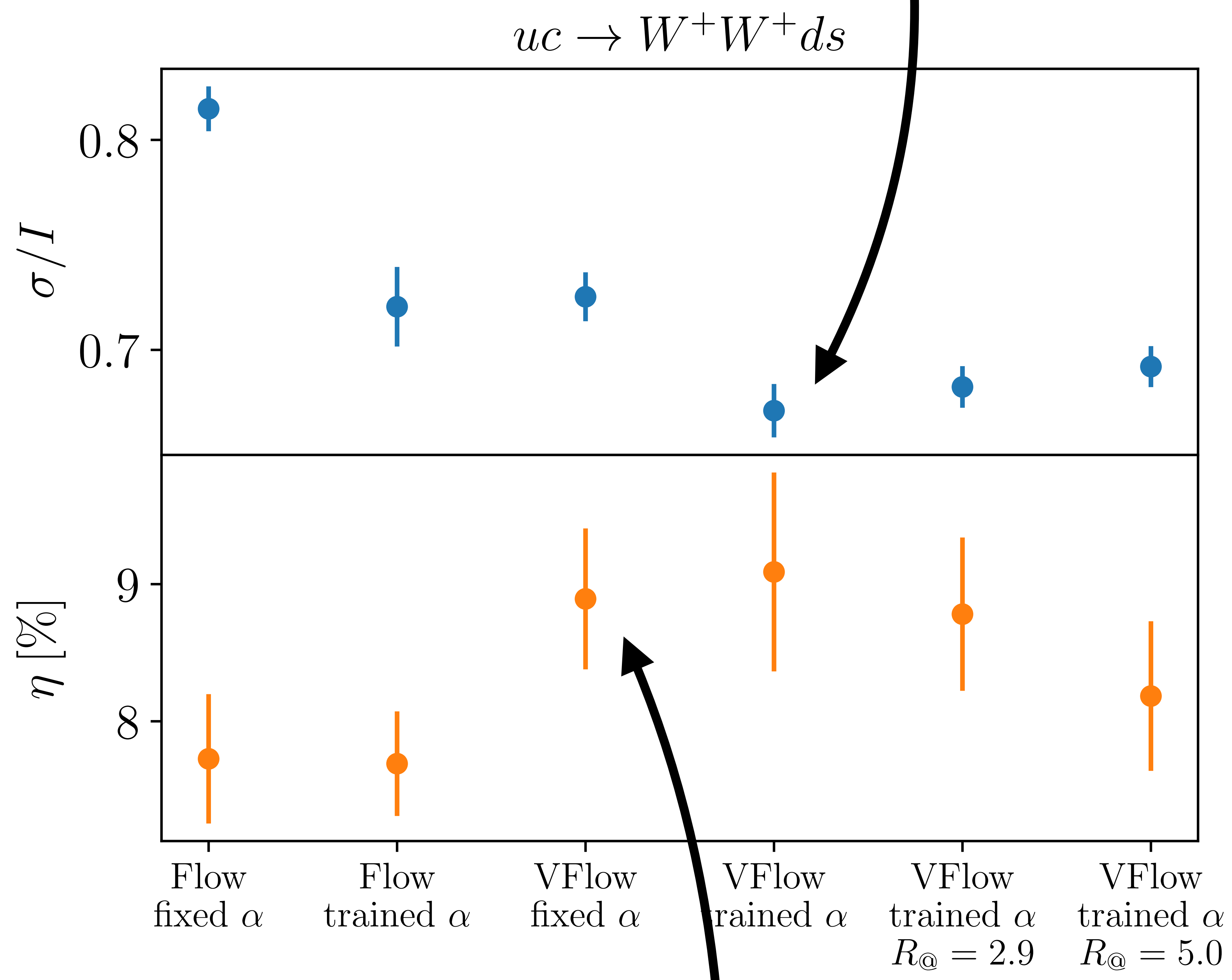
(preliminary)

LHC Example: Vector Boson Scattering



Unweighting efficiency improved up to factor ~9 compared to VEGAS

Significant improvement from trained channel weights

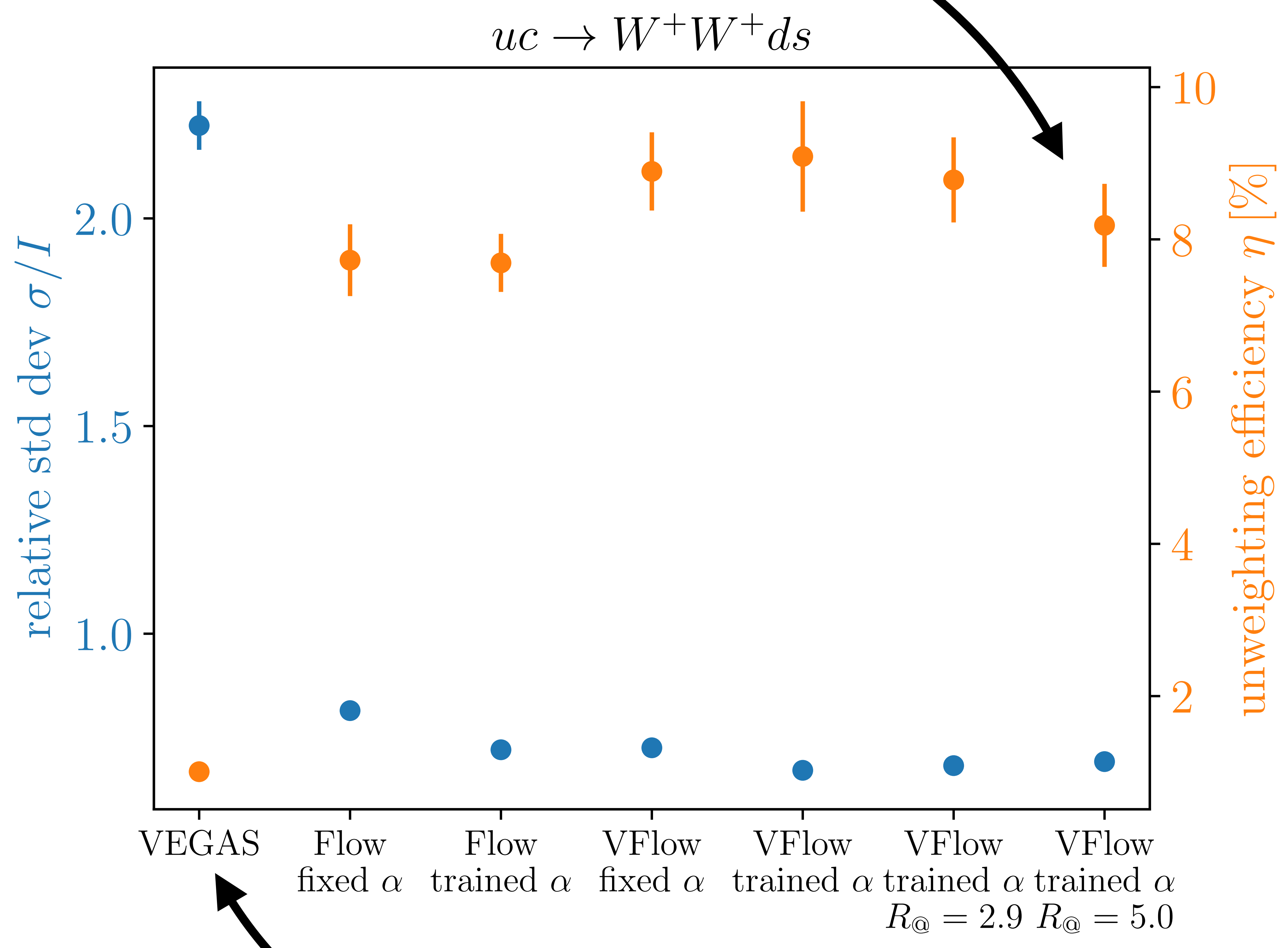


Big improvement from VEGAS initialization

(preliminary)

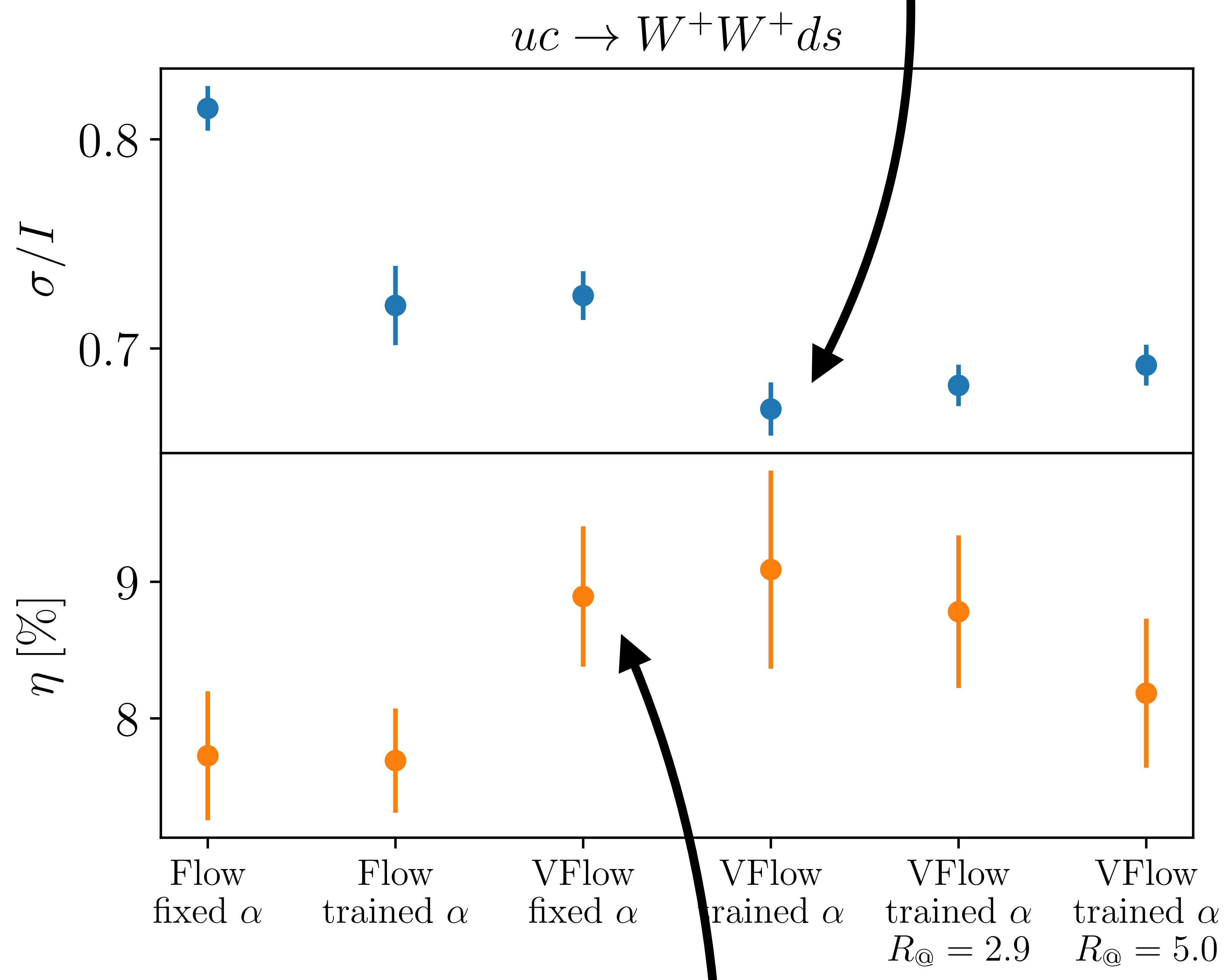
LHC Example: Vector Boson Scattering

Buffered training: small effect on performance, much faster training



Unweighting efficiency improved up to factor ~9 compared to VEGAS

Significant improvement from trained channel weights

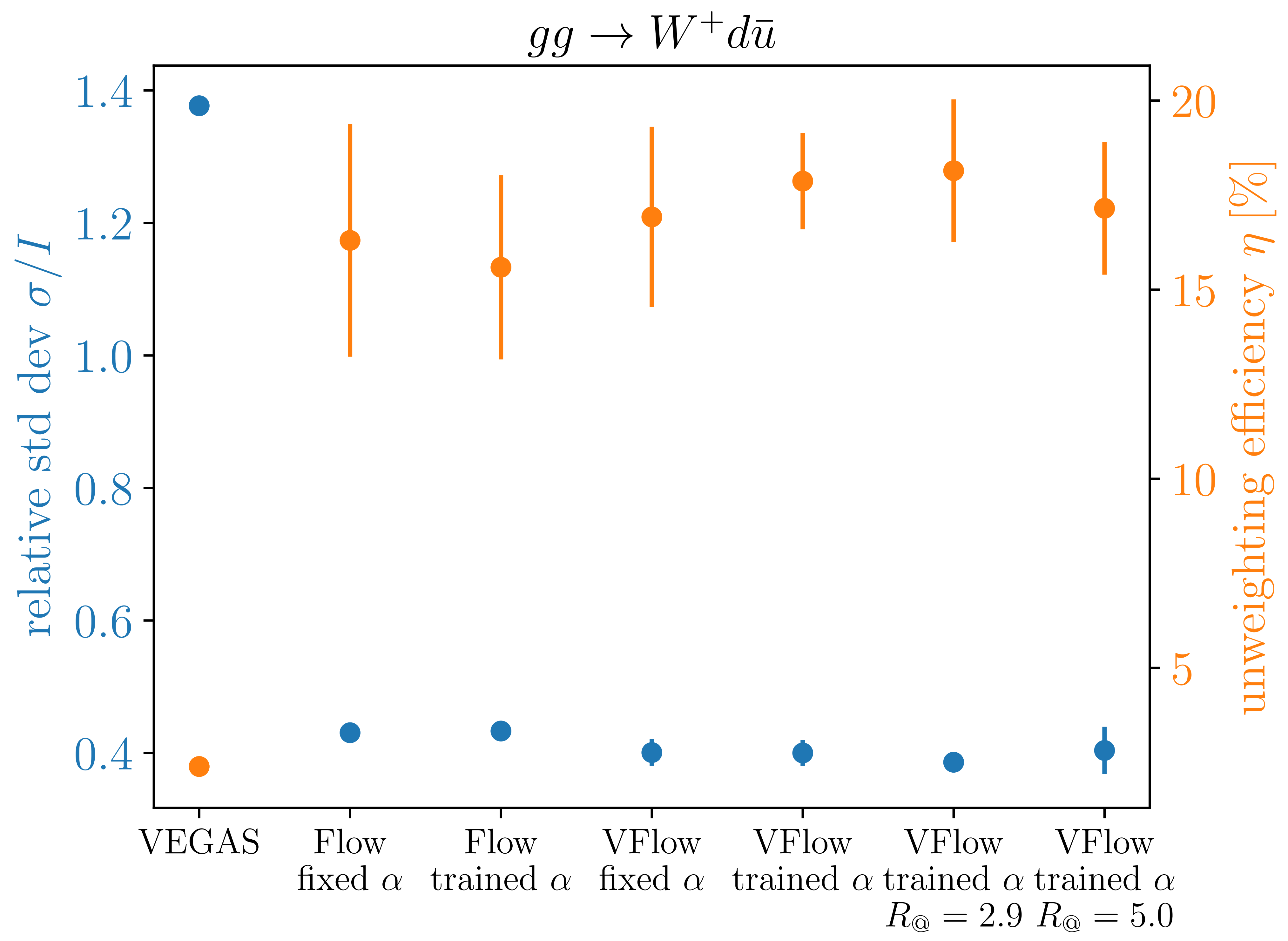


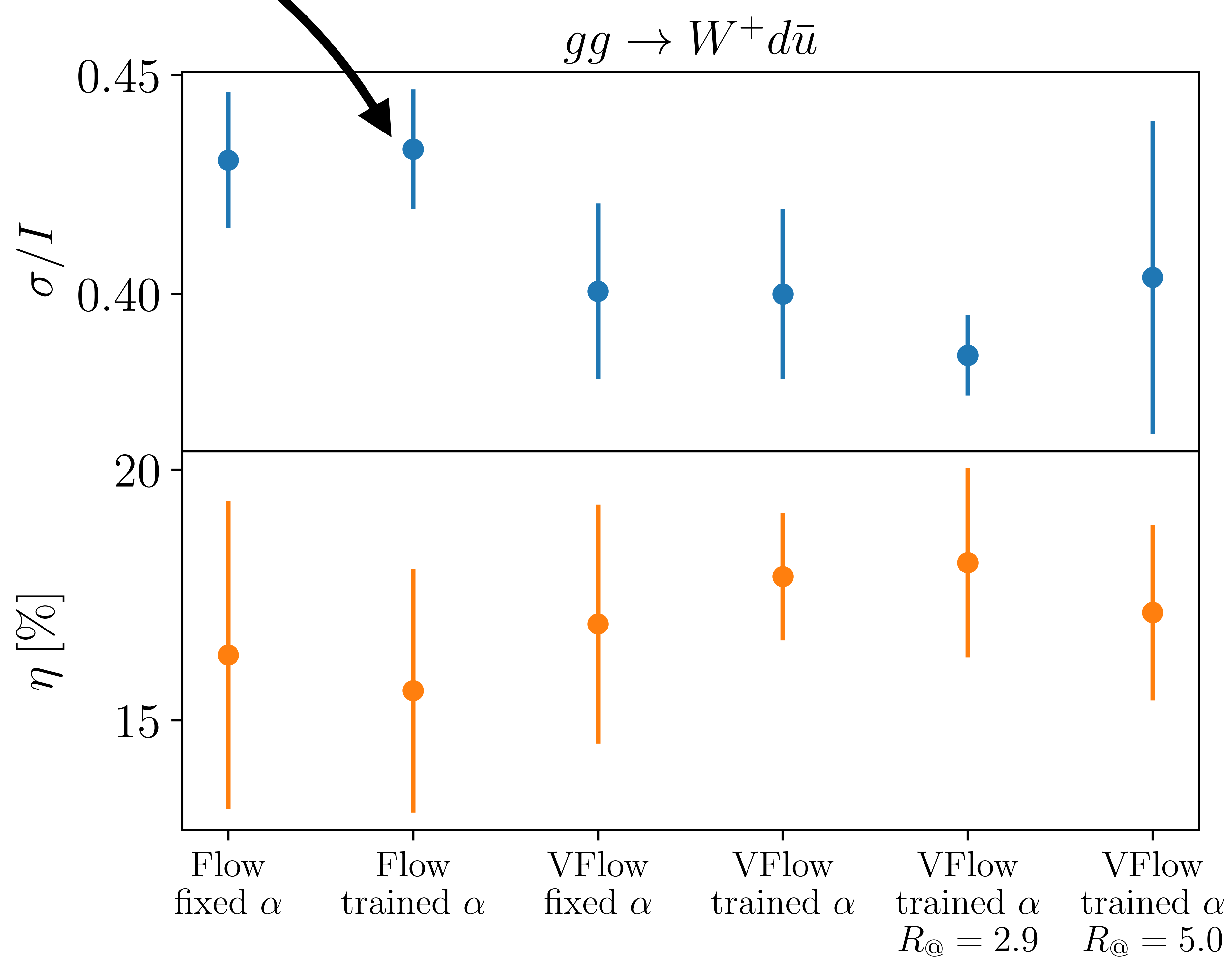
Big improvement from VEGAS initialization

(preliminary)

LHC Example: W + 2 jets

Process has small interference terms
→ no significant improvement from trained channel weights





(preliminary)

Otherwise similar to results for VBS

Outlook

Upcoming paper

Detailed comparison between MadNIS and standard MadGraph

→ more LHC processes

→ scaling with jet multiplicity

→ runtime comparison

→ test MadNIS features

Stay tuned!

Outlook

Upcoming paper

Detailed comparison between MadNIS and standard MadGraph

→ more LHC processes

→ scaling with jet multiplicity

→ runtime comparison

→ test MadNIS features

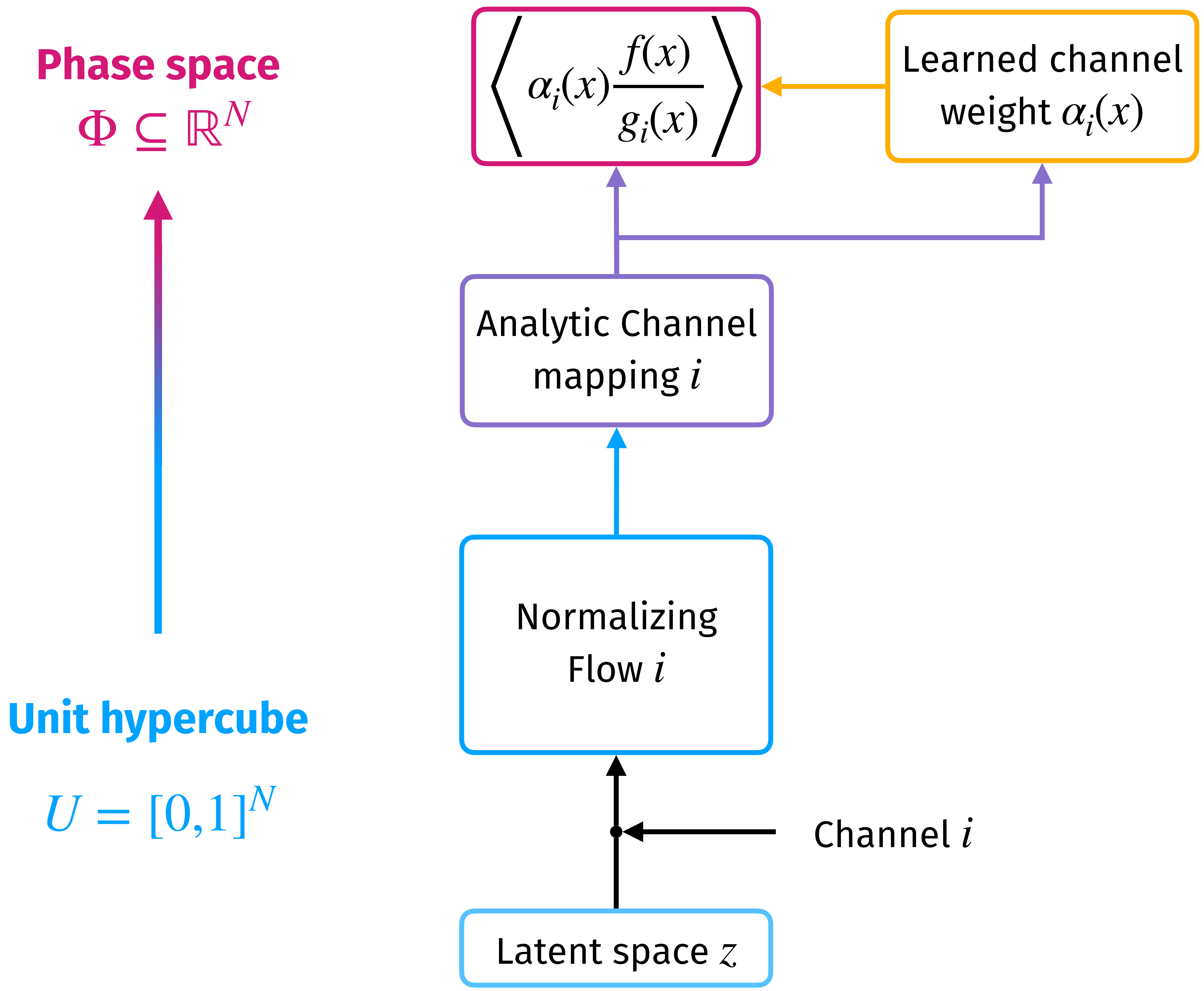
Stay tuned!

Future plans

Make MadNIS part of future MadGraph releases

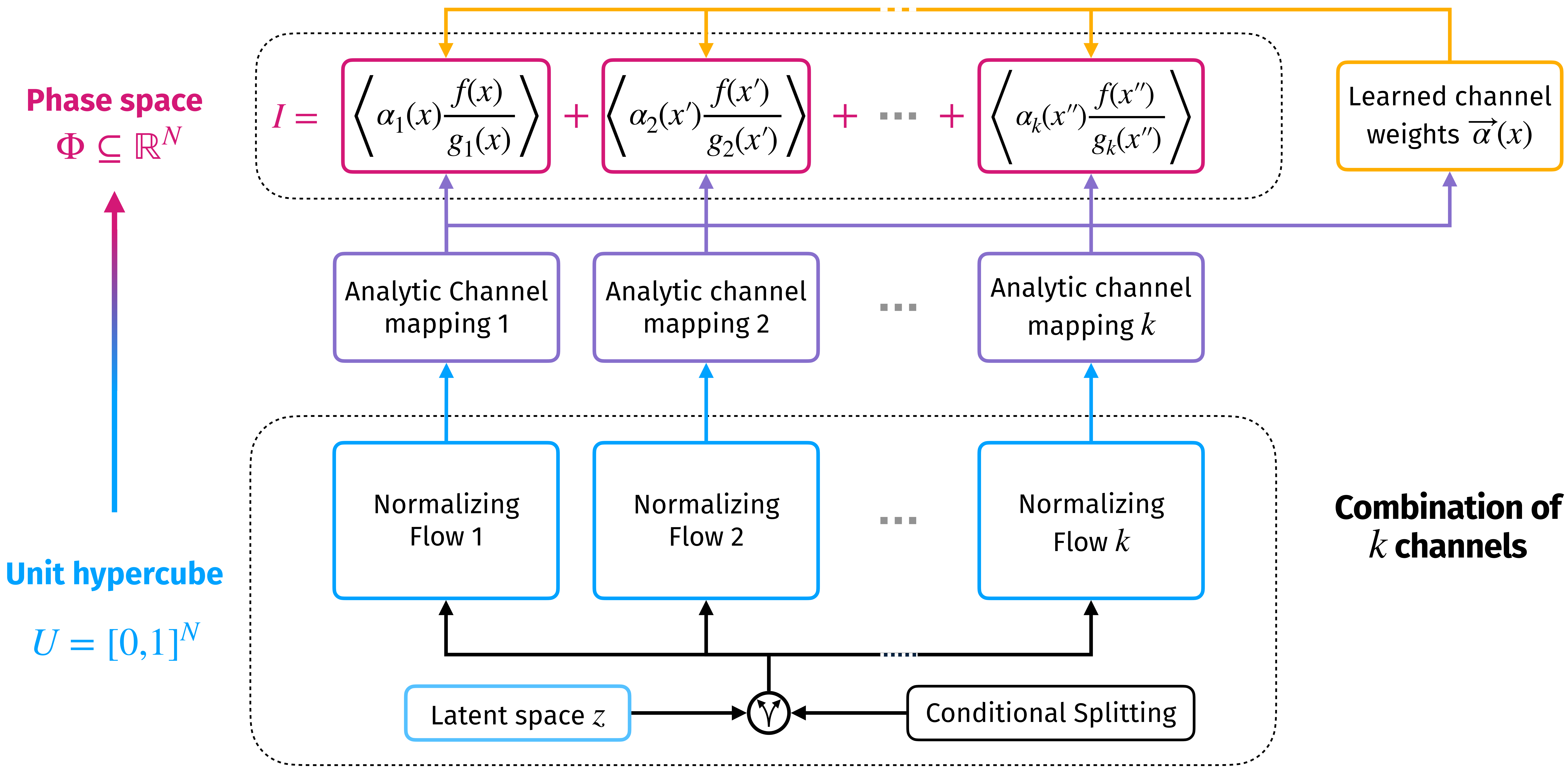
Appendix

MadNIS: Neural Importance Sampling

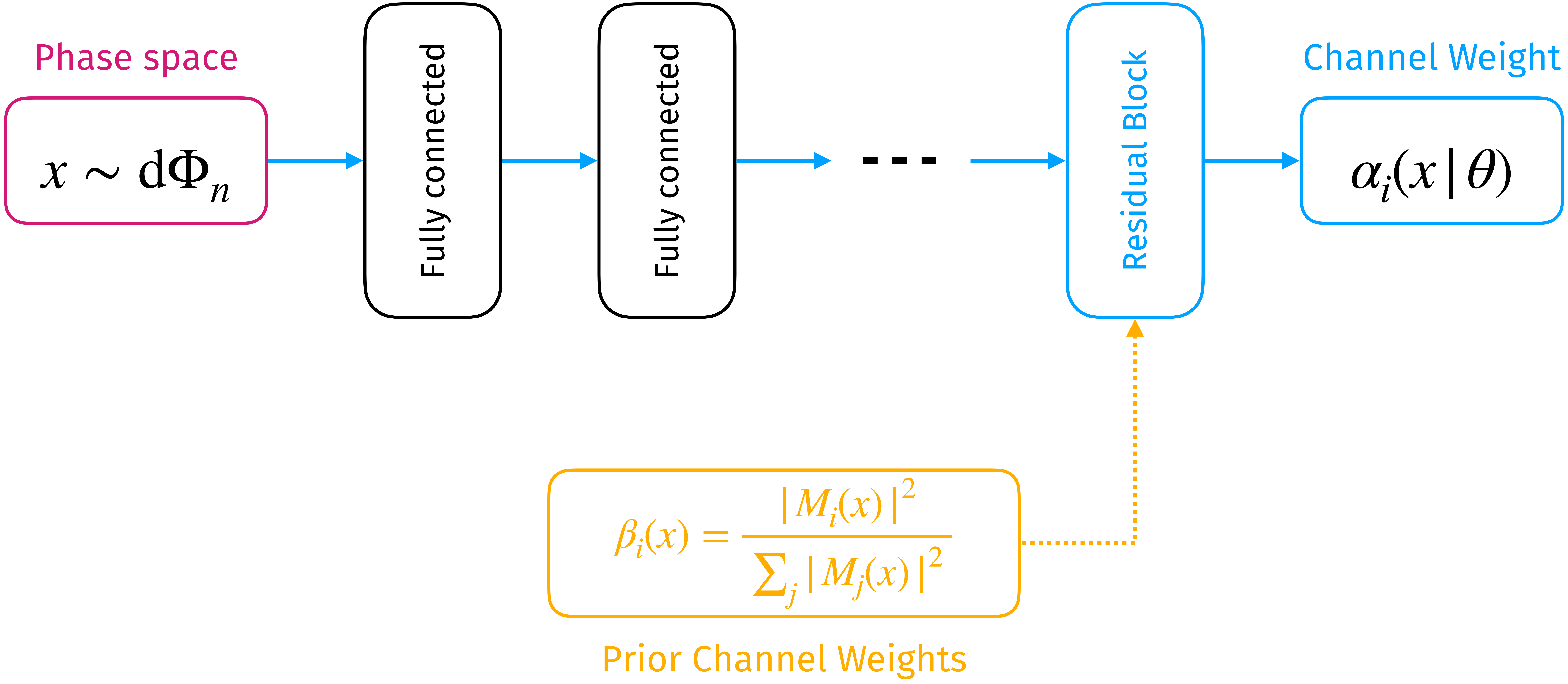


Single channel i

MadNIS: Neural Importance Sampling



Neural Channel Weights



Neural Channel Weights

Residual Block

Add prior

$$\alpha_i(x \mid \theta) = \log \beta_i(x) + \Delta_i(x \mid \theta)$$

Normalization

$$\alpha_i(x \mid \theta) \rightarrow \hat{\alpha}_i(x \mid \theta) = \frac{\beta_i(x) \exp \Delta_i(x \mid \theta)}{\sum_j \beta_j(x) \exp \Delta_j(x \mid \theta)}$$

Residual Block

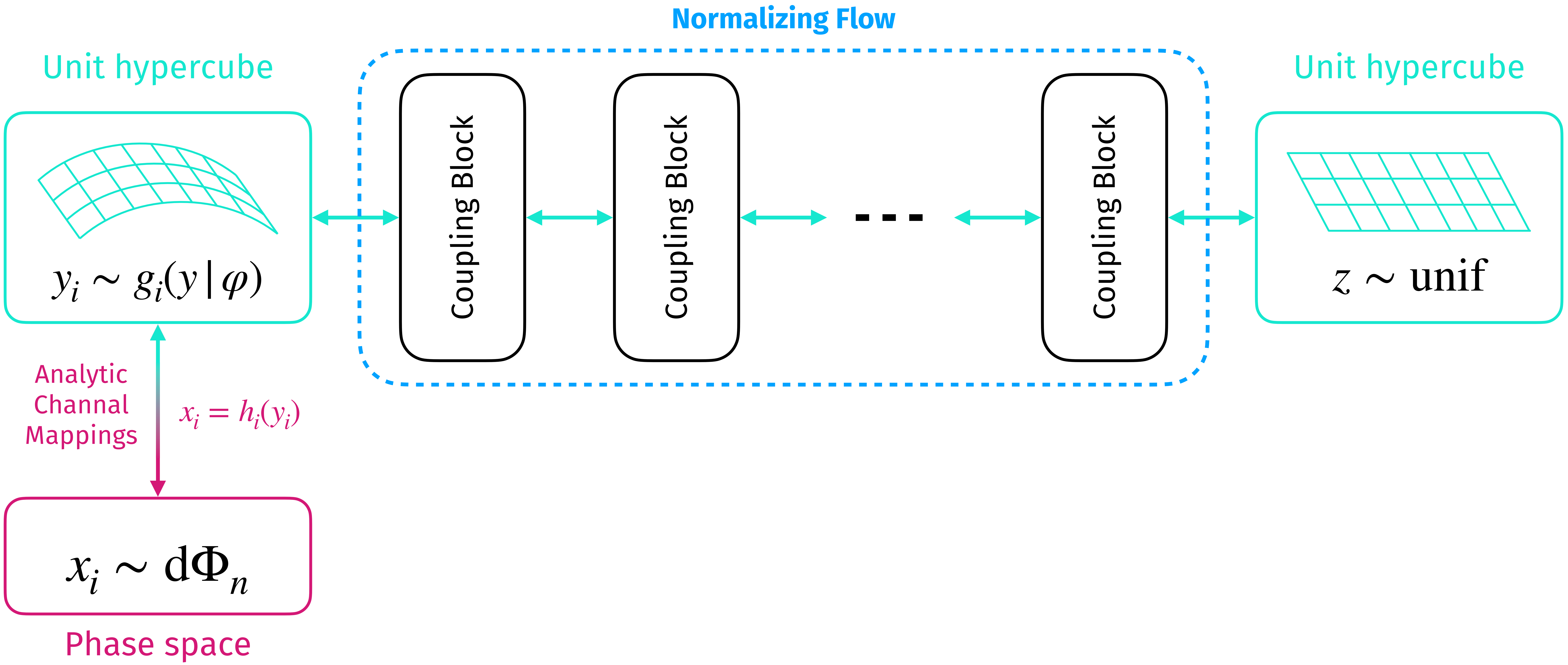
Channel Weight

$$\alpha_i(x \mid \theta)$$

$$\beta_i(x) = \frac{|M_i(x)|^2}{\sum_j |M_j(x)|^2}$$

Prior Channel Weights

Neural Importance Sampling



SymFI Multi-Channel