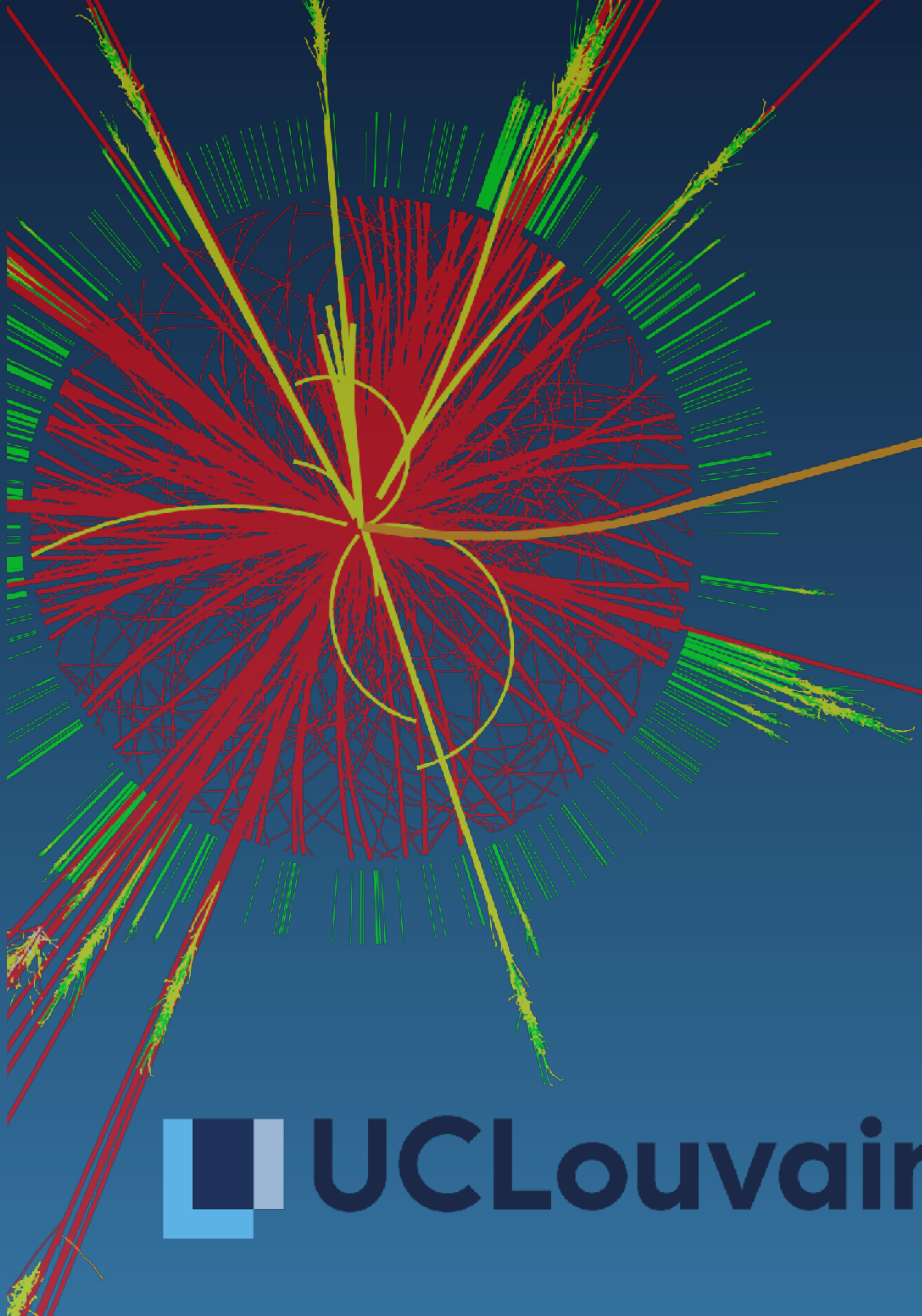


Machine Learning for Event Generation

Sapienta ex machina?

UCLouvain

MCnet School 2023 — Durham

Ramon Winterhalder — UC Louvain

Plan of attack

1. Machine learning for particle physics?

2. Normalizing flows

3. MadNIS

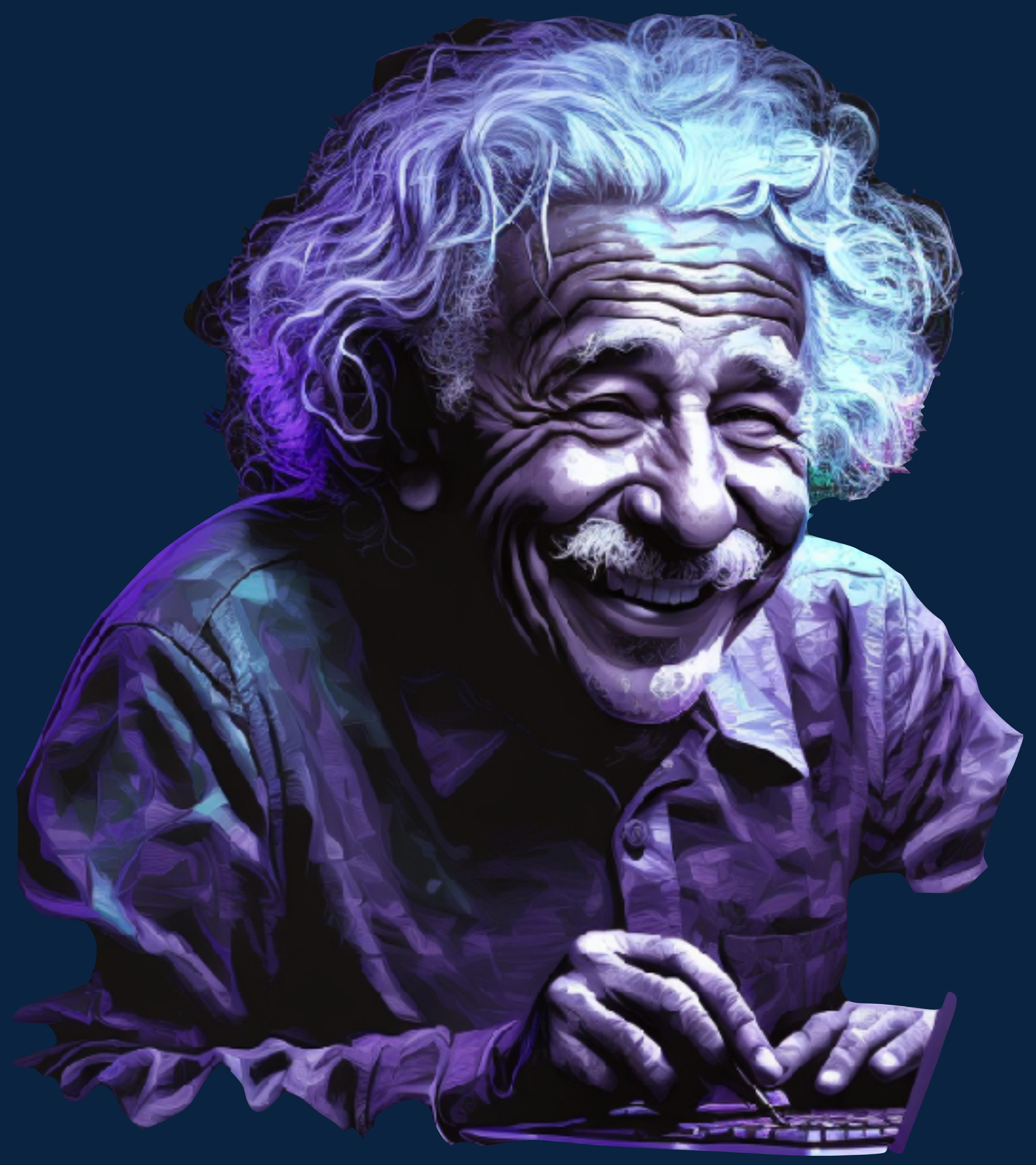
4. Summary and discussion

Part I

Machine learning for particle physics?

Why care about ML in particle physics?

Why do we care about ML?

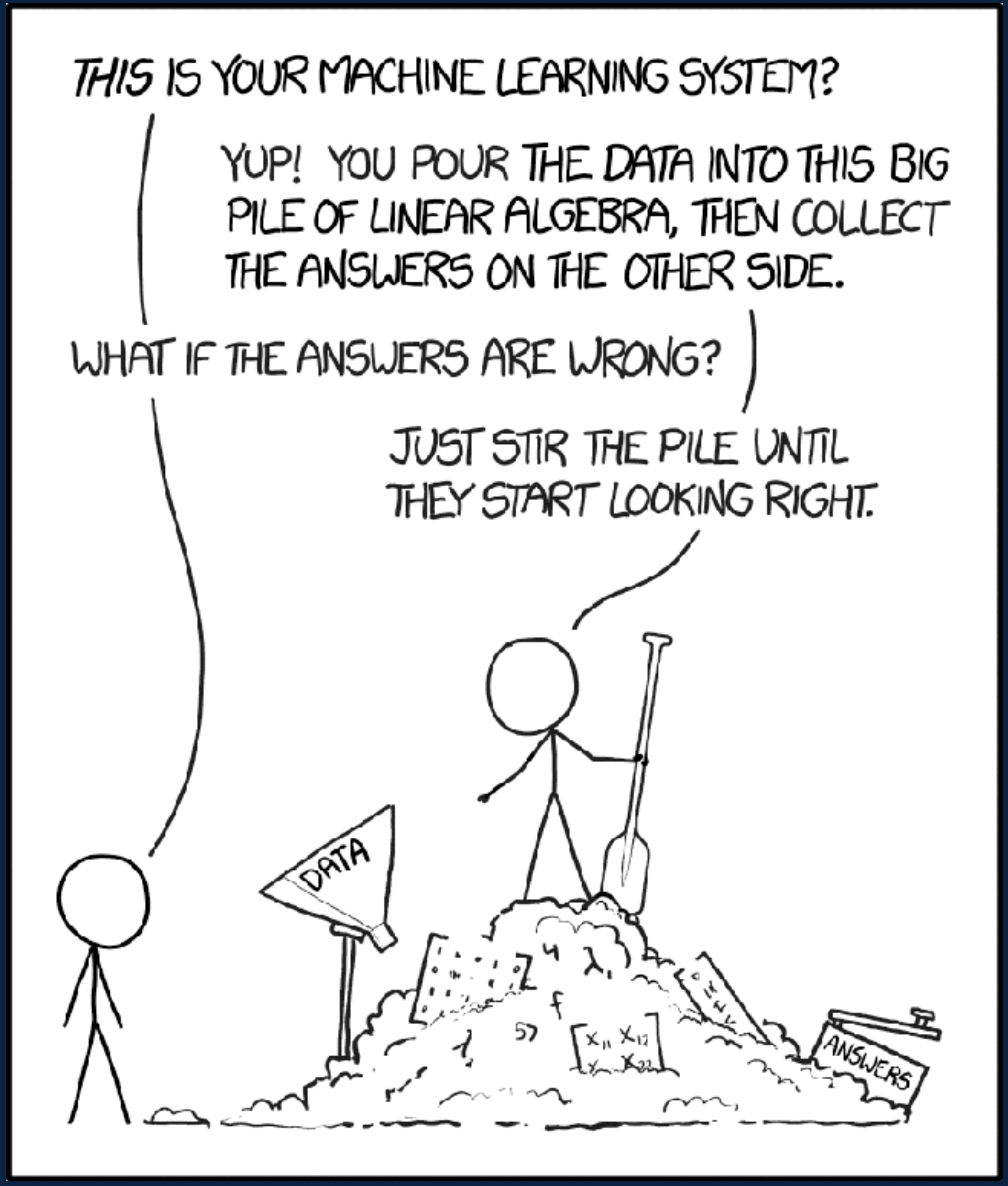


> “We care about machine learning because it can improve data analysis, simulation and modeling, lead to new discoveries, and foster cross-disciplinary collaboration
>
> **ChatGPT**

How machine learning often feels like

Aim of this lecture:

Giving you the ideas to use it right!



How machine learning often feels like

Aim of this lecture:

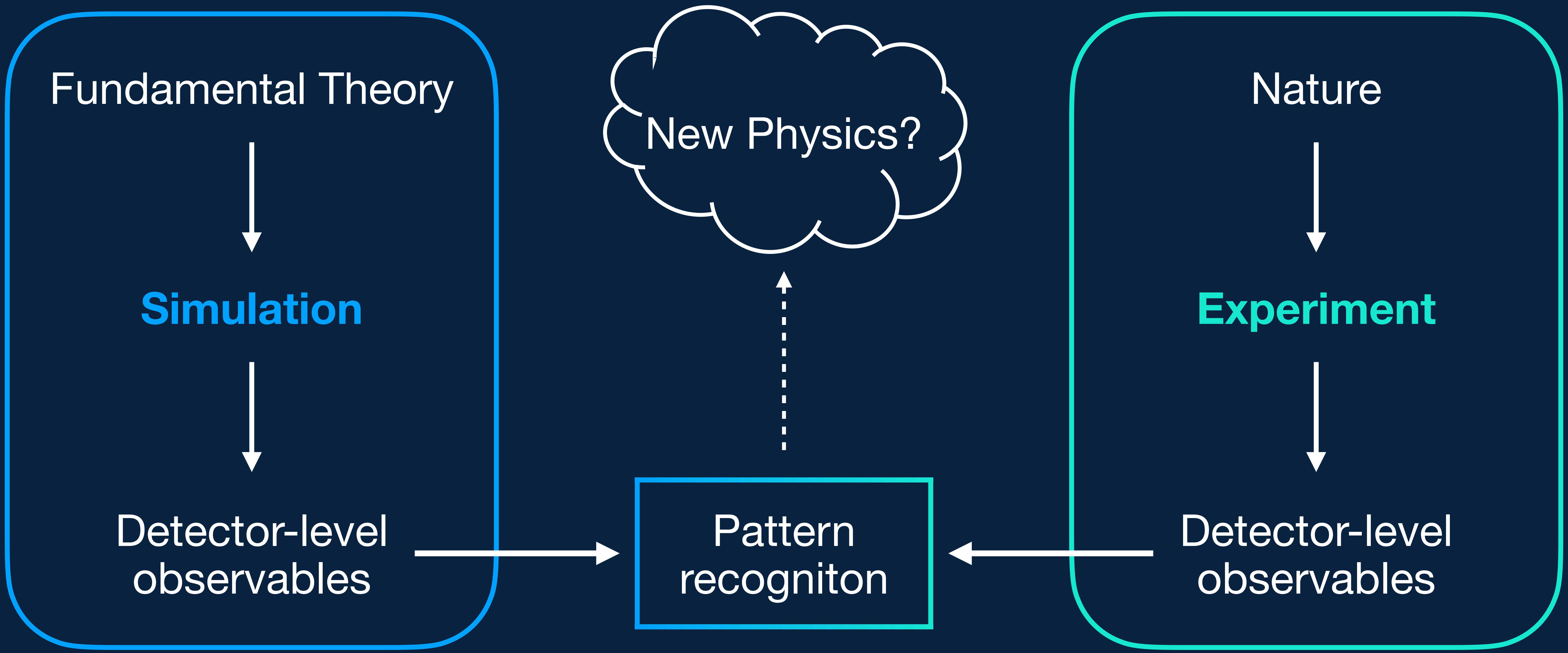
Giving you the ideas to use it right!

Be aware!

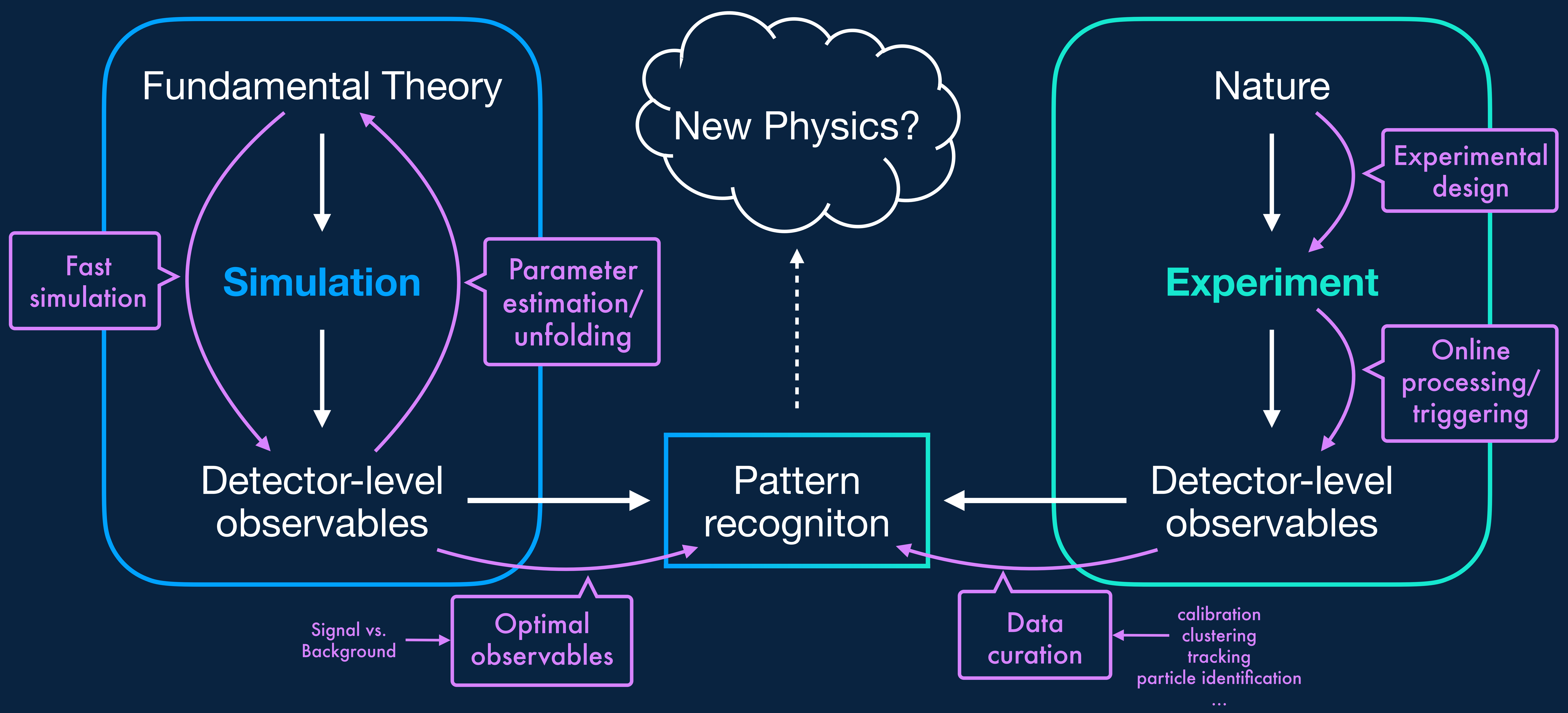
The core of machine learning is to find structure in data - **no more no less!**

Thanks to machine-learning algorithms, the robot apocalypse was short-lived.

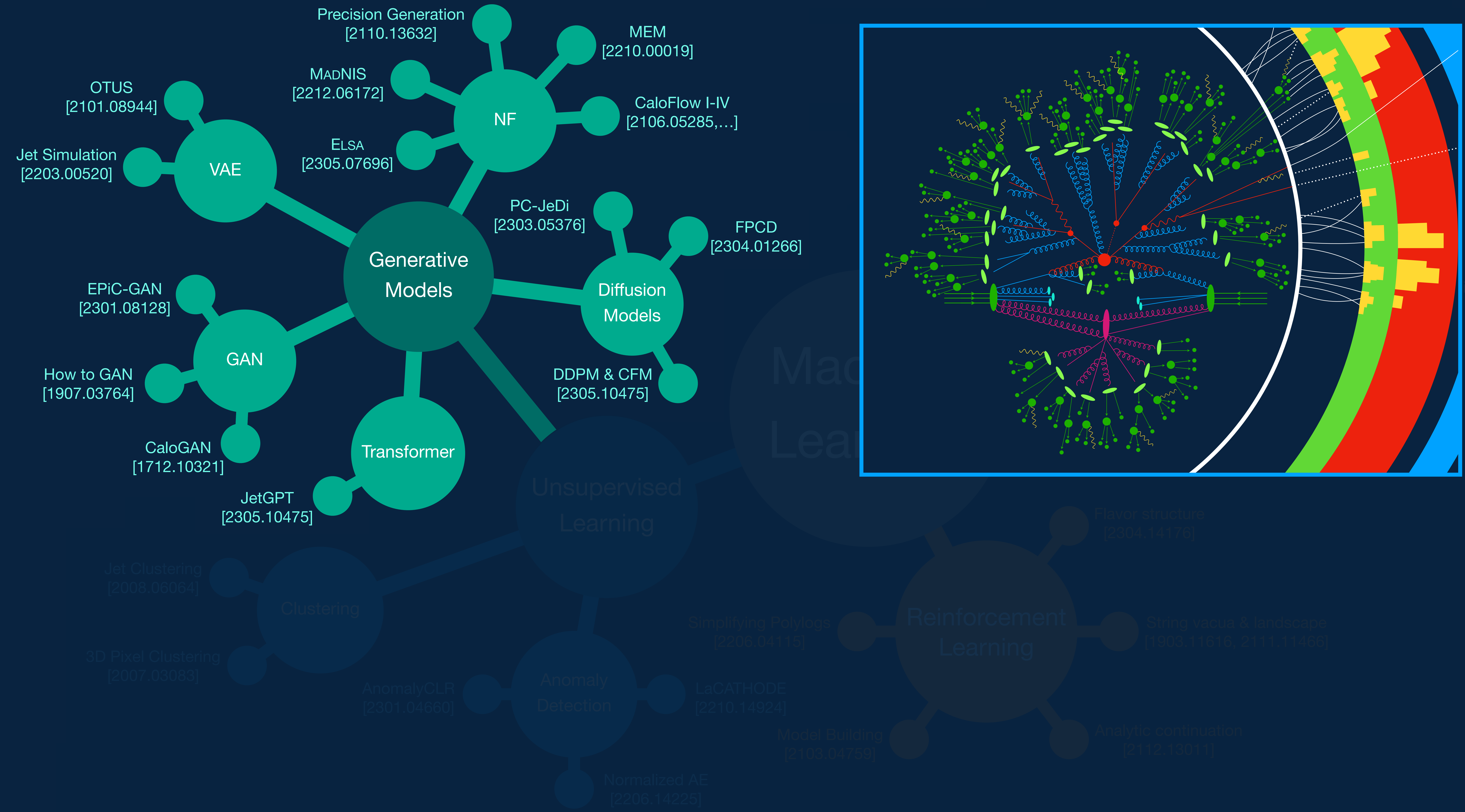
LHC analysis (oversimplified)



LHC analysis + ML

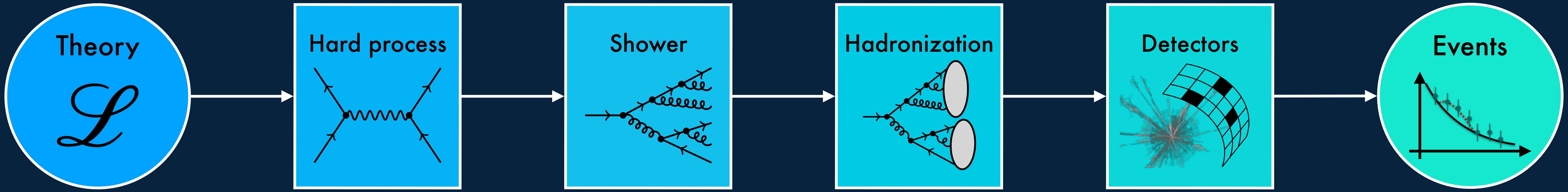


Machine Learning
Supervised Learning
Classification
Point Clouds [2102.05073]
PELICAN [2211.00454]
Energy Flow Networks [1810.05165]
ParticleNet [1902.08570]
Bayesian Tagger [1904.10004]
Regression
Bayesian Regression [2003.11099]
NNPDF [2109.02653]
Matrix Elements [2206.14831]
Symbolic regression [2109.10414]
Reinforcement Learning
Flavor structure [2304.14176]
Simplifying Polylogs [2206.04115]
String vacua & landscape [1903.11616, 2111.11466]
Model Building [2103.04759]
Analytic continuation [2112.13011]
Unsupervised Learning
Clustering
Jet Clustering [2008.06064]
3D Pixel Clustering [2007.03083]
Anomaly Detection
AnomalyCLR [2301.04660]
LaCATHODE [2210.14924]
Normalized AE [2206.14225]
Generative Models
VAE
OTUS [2101.08944]
Jet Simulation [2203.00520]
GAN
EPiC-GAN [2301.08128]
How to GAN [1907.03764]
CaloGAN [1712.10321]
Transformer
JetGPT [2305.10475]
NF
Precision Generation [2110.13632]
MEM [2210.00019]
MADNIS [2212.06172]
CaloFlow I-IV [2106.05285,...]
ELSA [2305.07696]
Diffusion Models
PC-JeDi [2303.05376]
FPCD [2304.01266]
DDPM & CFM [2305.10475]

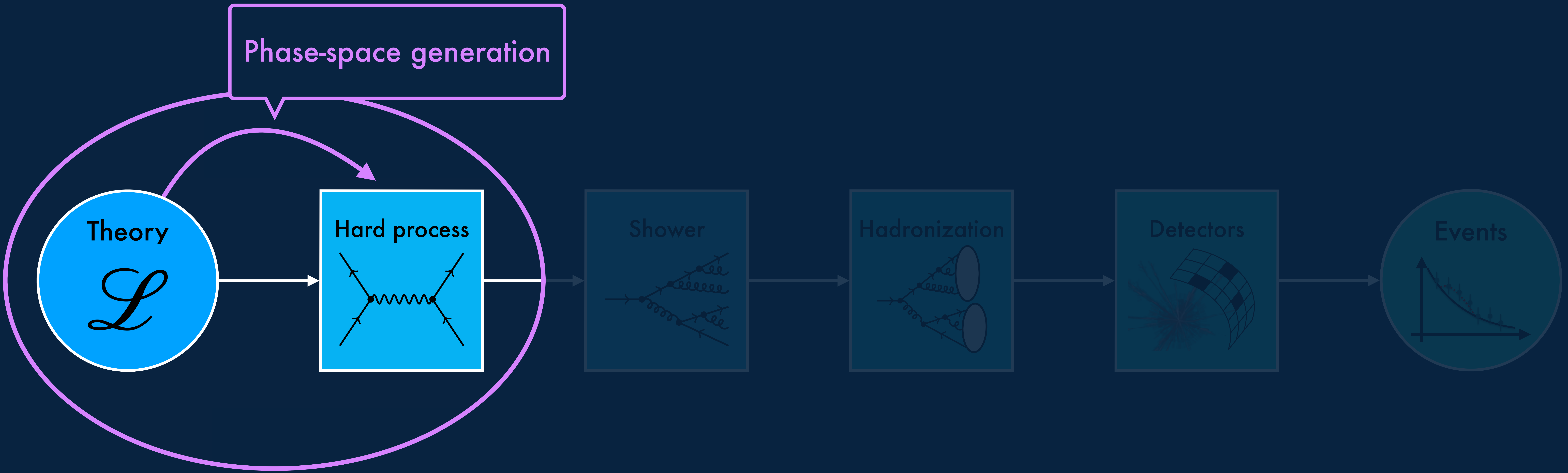

Generative Models
NF
Precision Generation [2110.13632]
MEM [2210.00019]
MADNIS [2212.06172]
CaloFlow I-IV [2106.05285,…]
ELSA [2305.07696]
VAE
OTUS [2101.08944]
Jet Simulation [2203.00520]
Diffusion Models
PC-JeDi [2303.05376]
FPCD [2304.01266]
DDPM & CFM [2305.10475]
GAN
EPiC-GAN [2301.08128]
How to GAN [1907.03764]
CaloGAN [1712.10321]
Transformer
JetGPT [2305.10475]
Unsupervised Learning
Clustering
Jet Clustering [2008.06064]
3D Pixel Clustering [2007.03083]
Anomaly Detection
AnomalyCLR [2301.04660]
LaCATHODE [2210.14924]
Normalized AE [2206.14225]
Reinforcement Learning
Flavor structure [2304.14176]
Simplifying Polylogs [2206.04115]
String vacua & landscape [1903.11616, 2111.11466]
Model Building [2103.04759]
Analytic continuation [2112.13011]


Generative Models
VAE
OTUS [2101.08944]
Jet Simulation [2203.00520]
NF
Precision Generation [2110.13632]
MEM [2210.00019]
MADNIS [2212.06172]
CaloFlow I-IV [2106.05285,…]
ELSA [2305.07696]
Diffusion Models
PC-JeDi [2303.05376]
FPCD [2304.01266]
DDPM & CFM [2305.10475]
GAN
EPiC-GAN [2301.08128]
How to GAN [1907.03764]
CaloGAN [1712.10321]
Transformer
JetGPT [2305.10475]
?
Theory
$\mathscr{L}$
Hard process
Shower
Hadronization
Detectors
Events

ML improved simulations

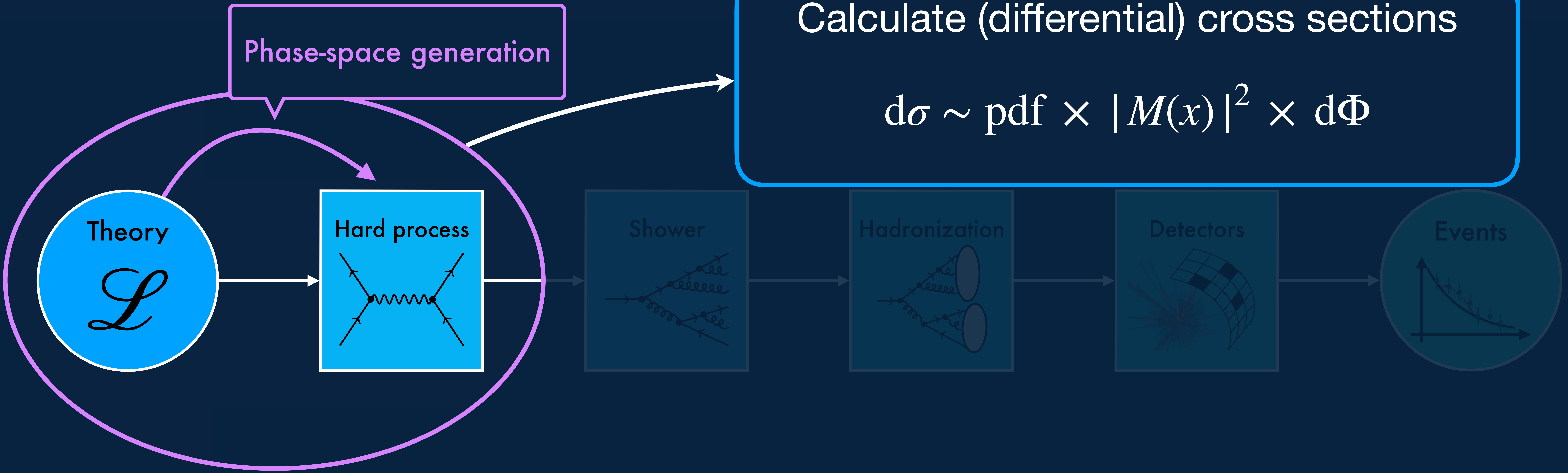


ML improved simulations



BDT [1707.00028, …], NN [1810.11509, 2009.07819, …]
NF [2001.05486, 2001.05478, 2001.10028, 2005.12719, 2112.09145, 2212.06172, …]

ML improved simulations

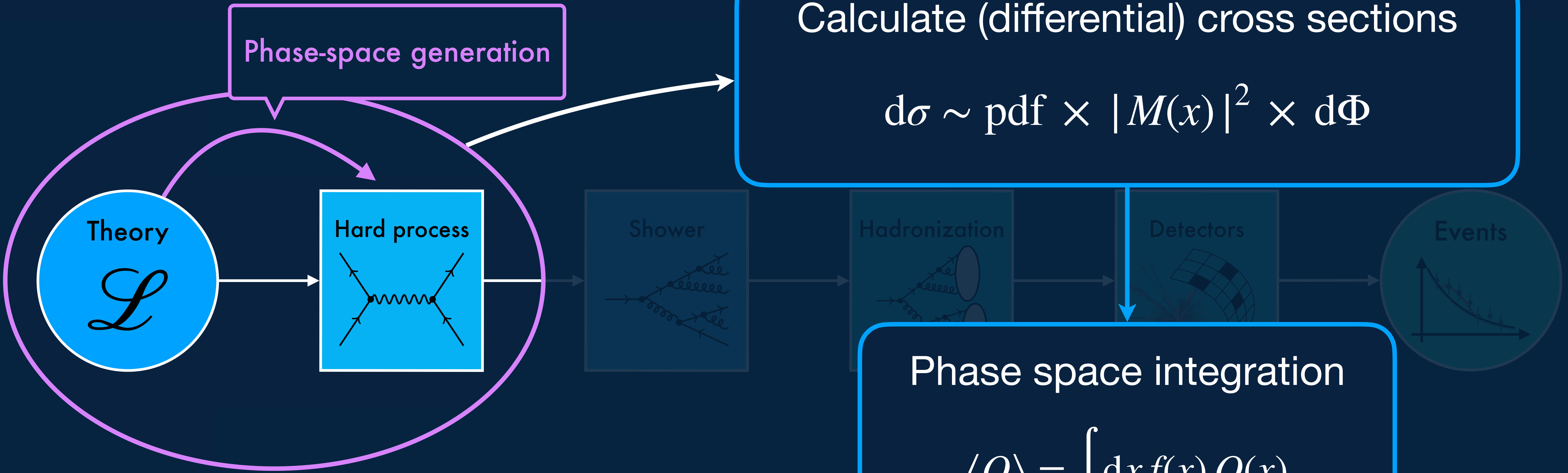


Calculate (differential) cross sections

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

BDT [1707.00028, …], NN [1810.11509, 2009.07819, …]
NF [2001.05486, 2001.05478, 2001.10028, 2005.12719, 2112.09145, 2212.06172, …]

ML improved simulations



Calculate (differential) cross sections

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

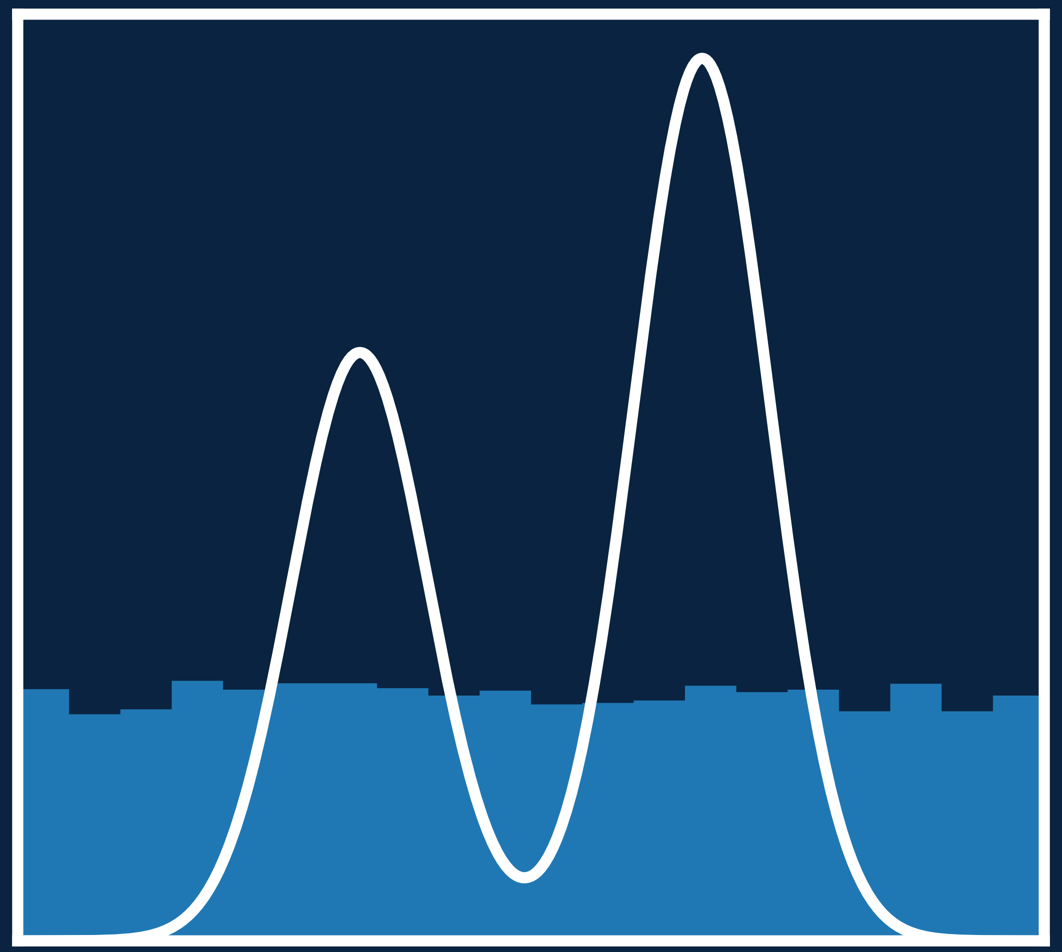
Phase space integration

$$\langle O \rangle = \int dx\, f(x)\, O(x)$$

BDT [1707.00028, …], NN [1810.11509, 2009.07819, …]
NF [2001.05486, 2001.05478, 2001.10028, 2005.12719, 2112.09145, 2212.06172, …]

Monte Carlo integration

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

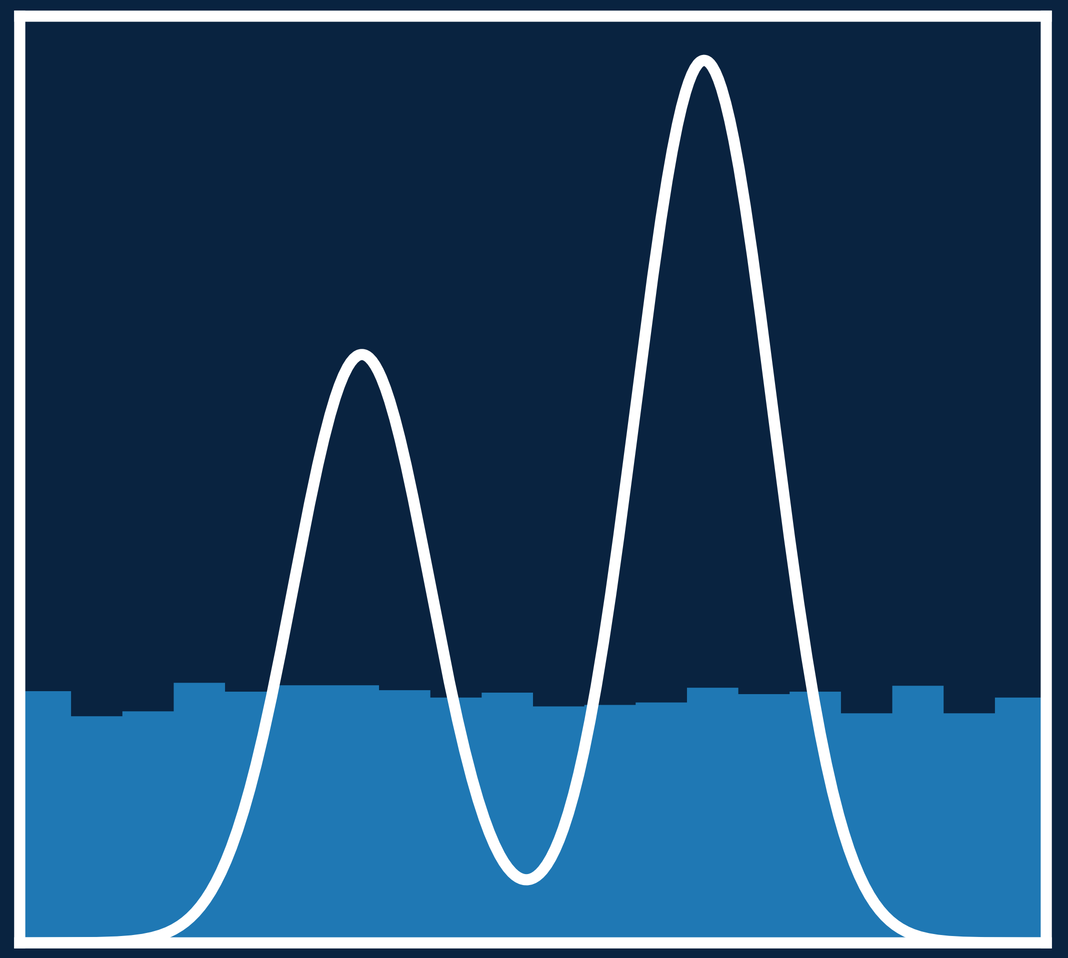


Flat sampling:
inefficient

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

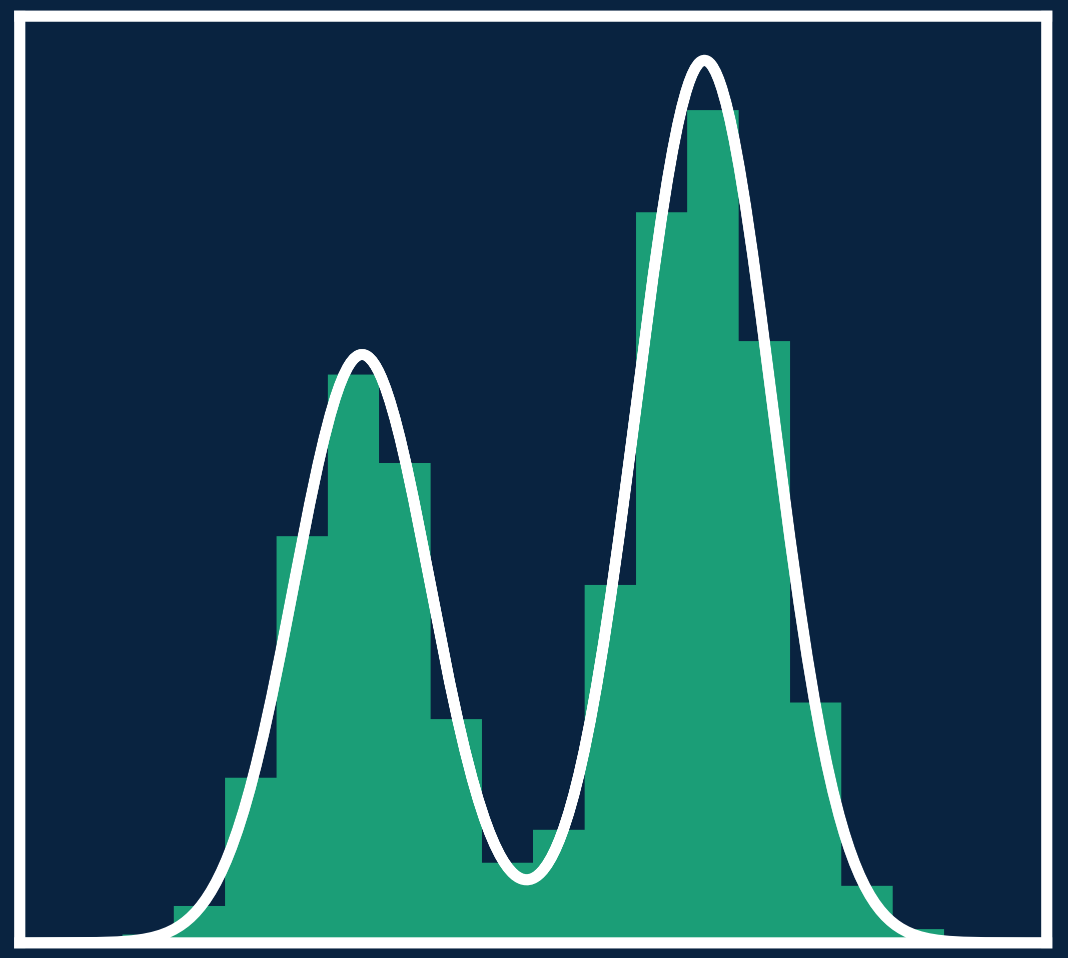
Monte Carlo integration

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



Flat sampling:
inefficient

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

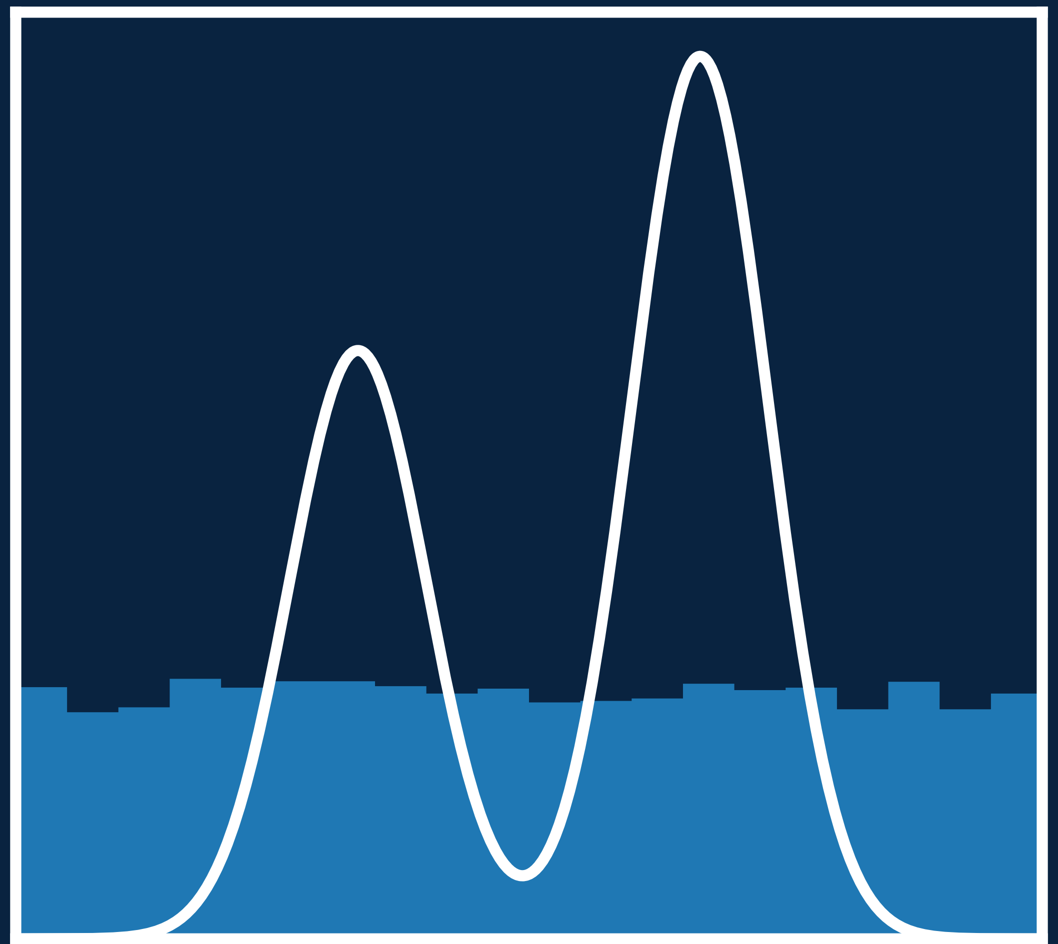


Importance sampling:
find g close to f

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

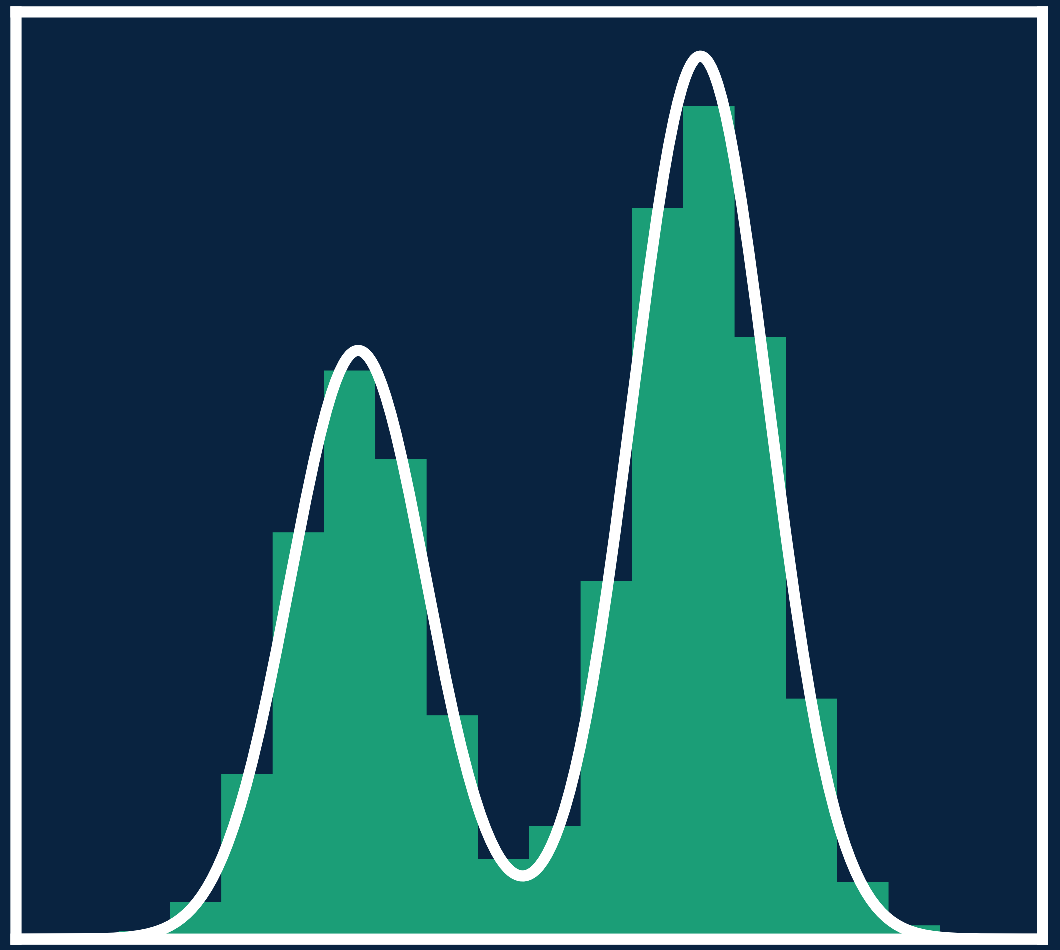
Monte Carlo integration

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



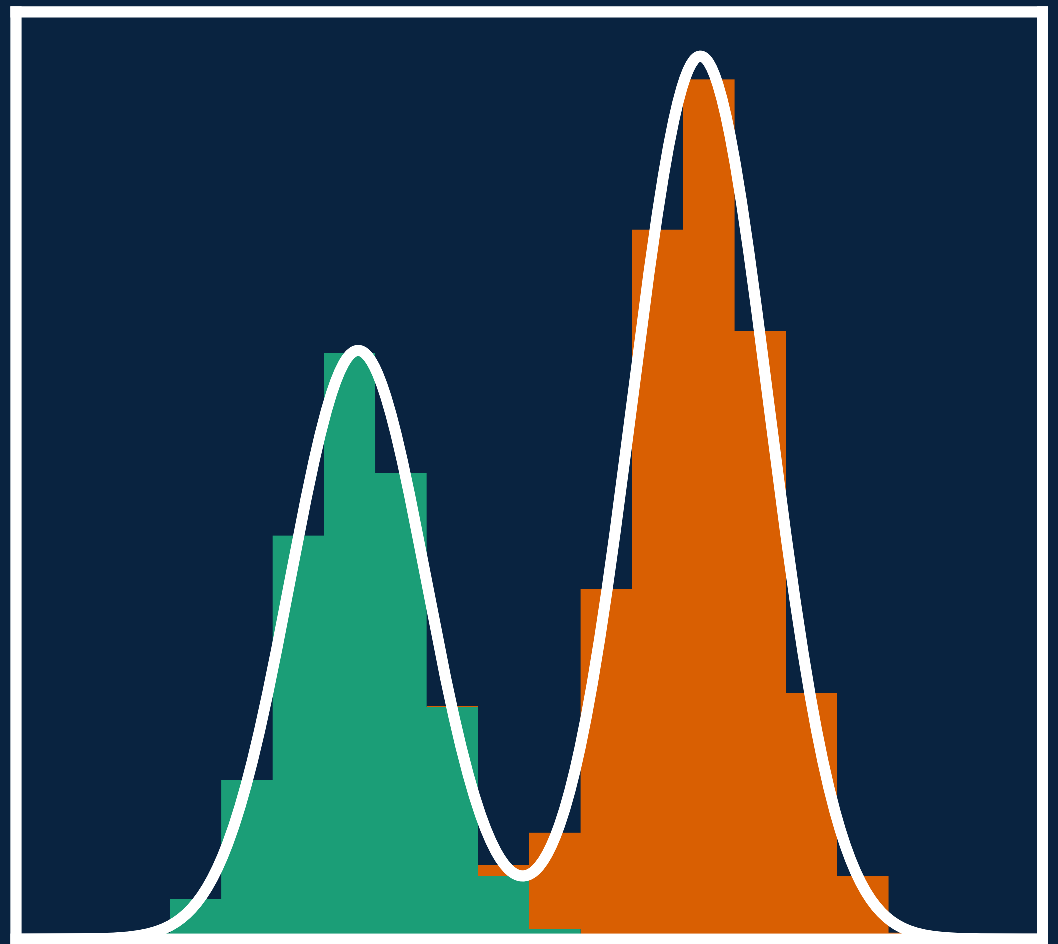
Flat sampling:
inefficient

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



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

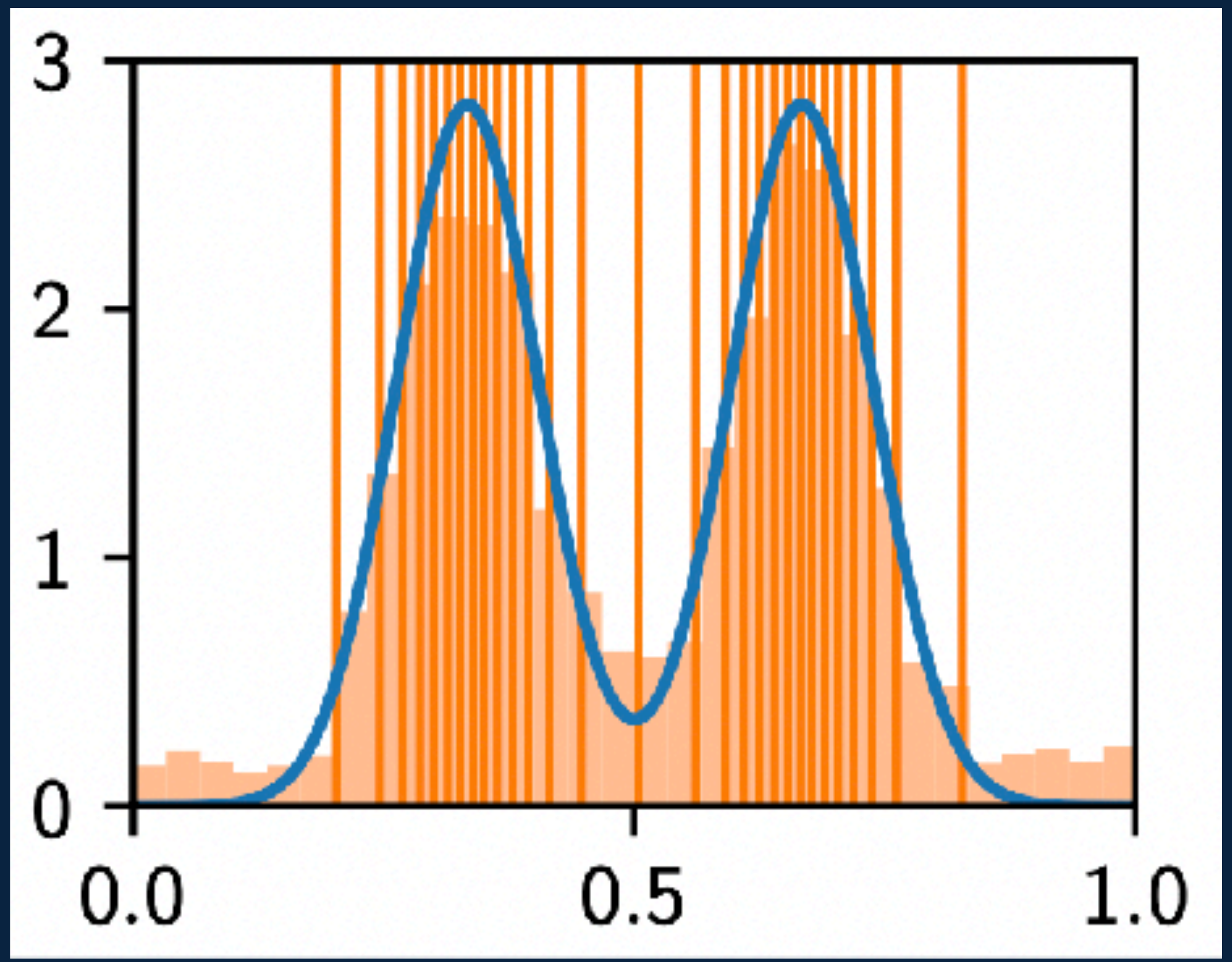
$$I = \sum_i \left\langle \alpha_i(x) \frac{f(x)}{g_i(x)} \right\rangle_{x \sim g_i(x)}$$

Importance sampling — VEGAS

Factorize probability

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

Fit bins with equal probability
and varying width

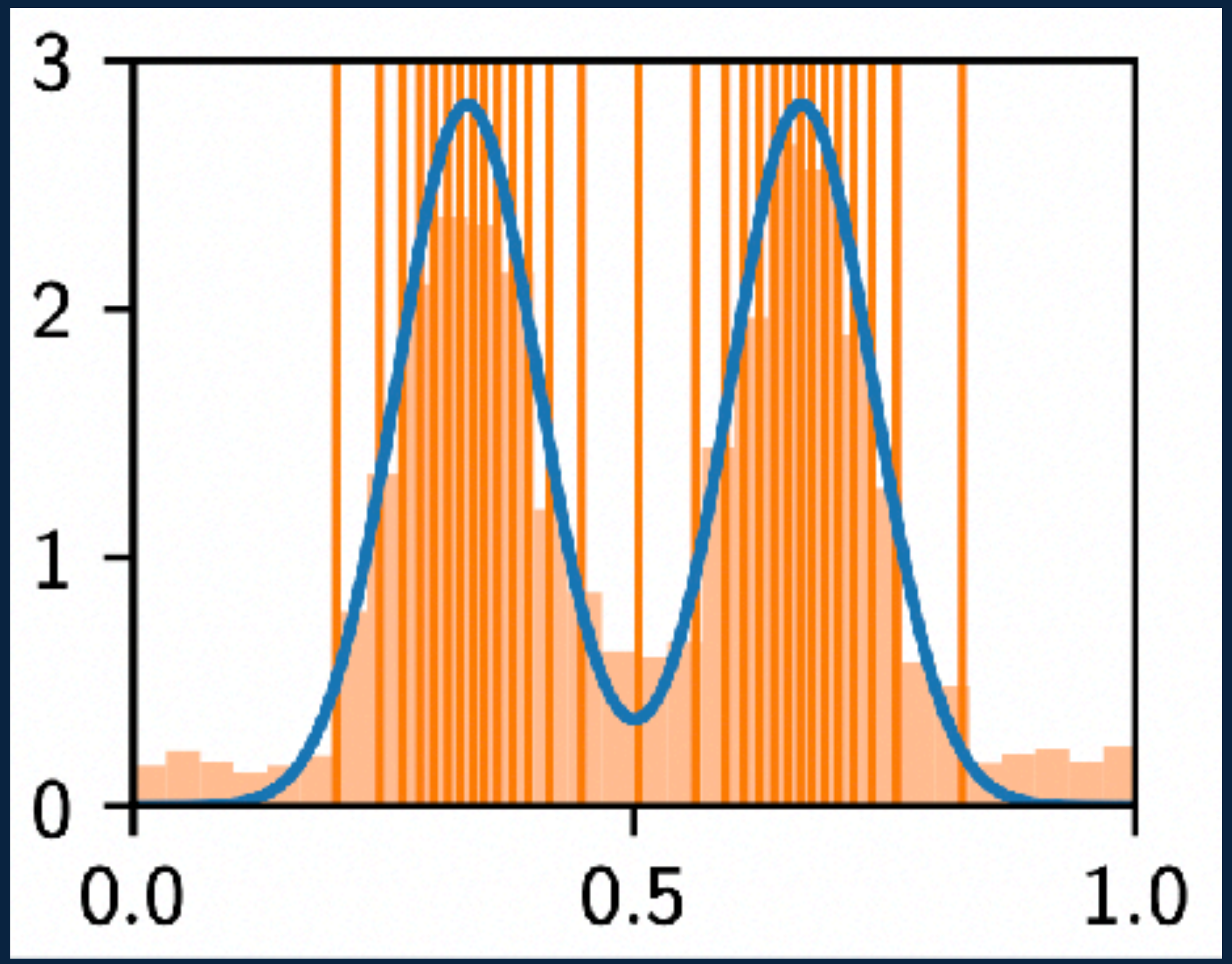


Importance sampling — VEGAS

Factorize probability

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

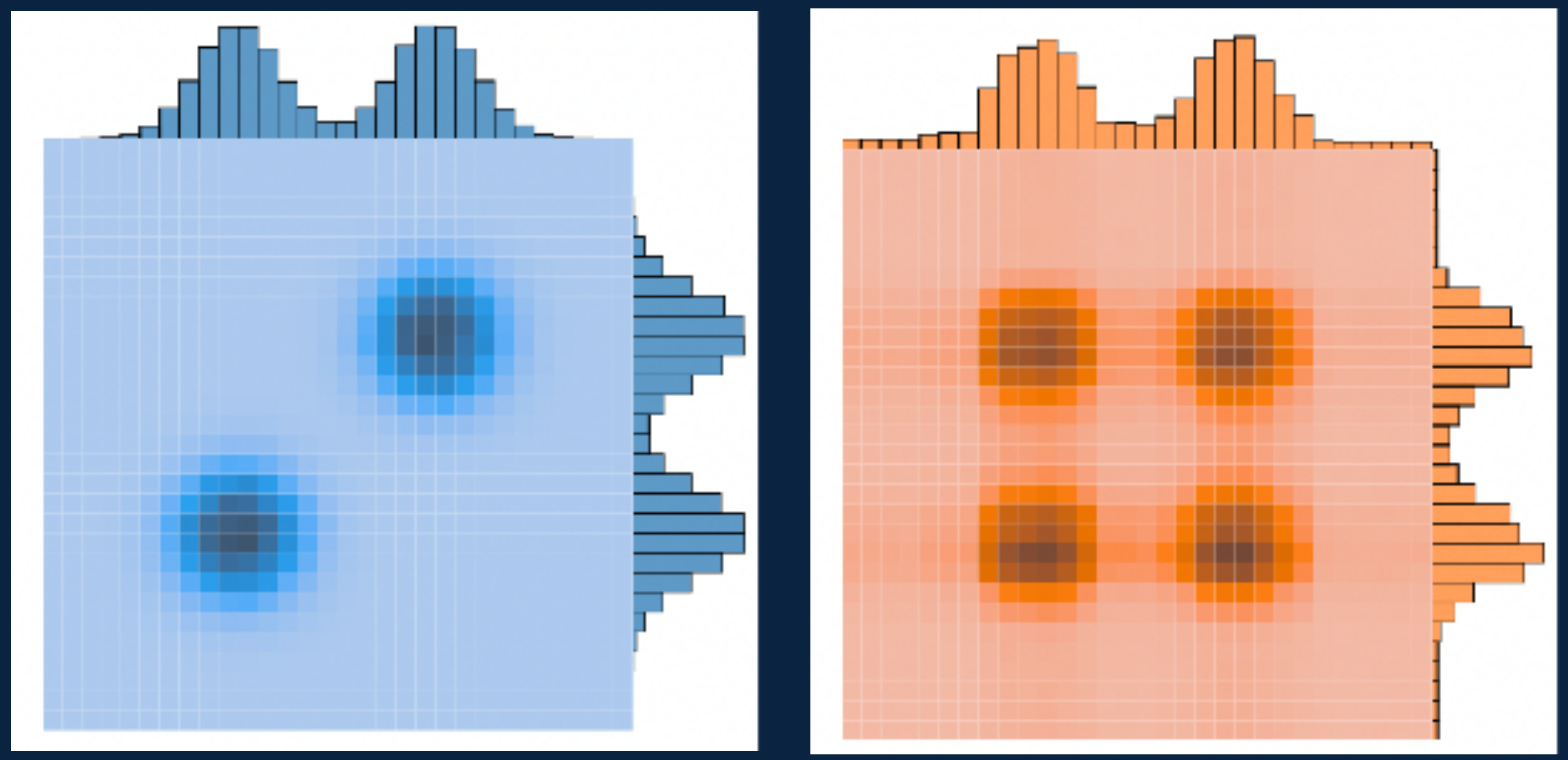
Fit bins with equal probability
and varying width



⊕ Computationally cheap

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

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

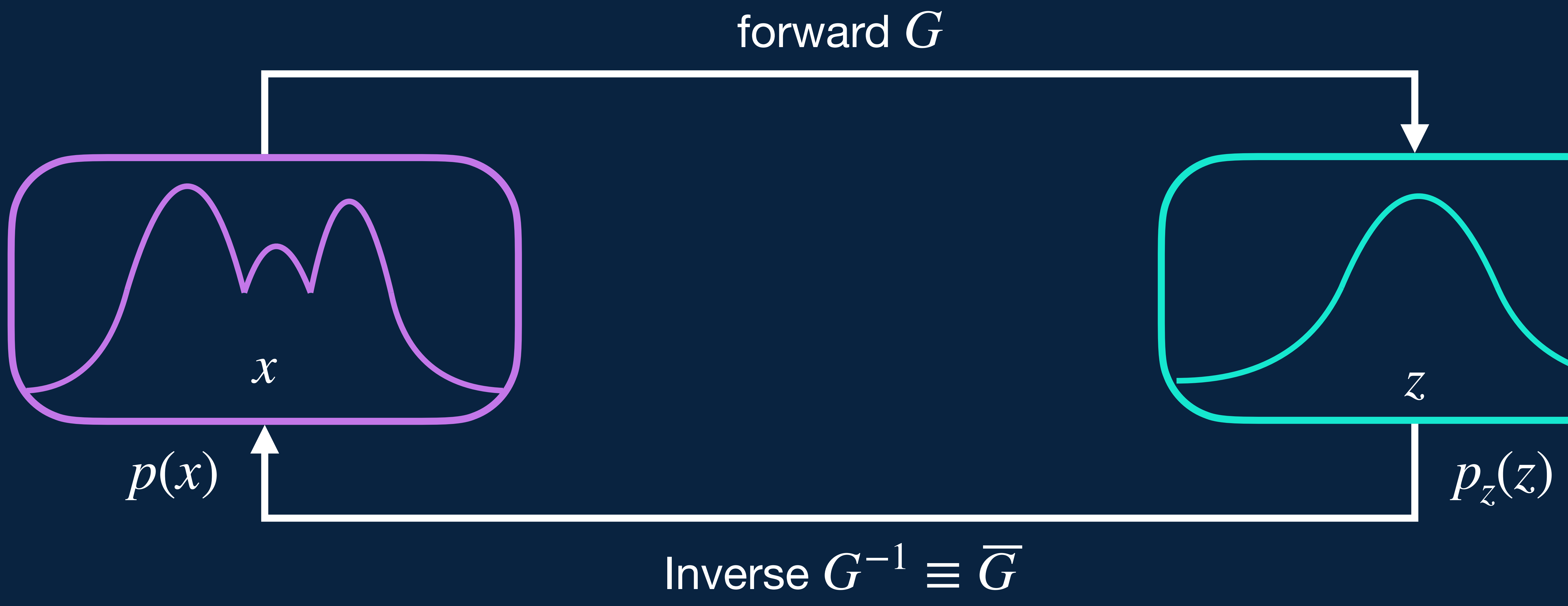


How can we do better?

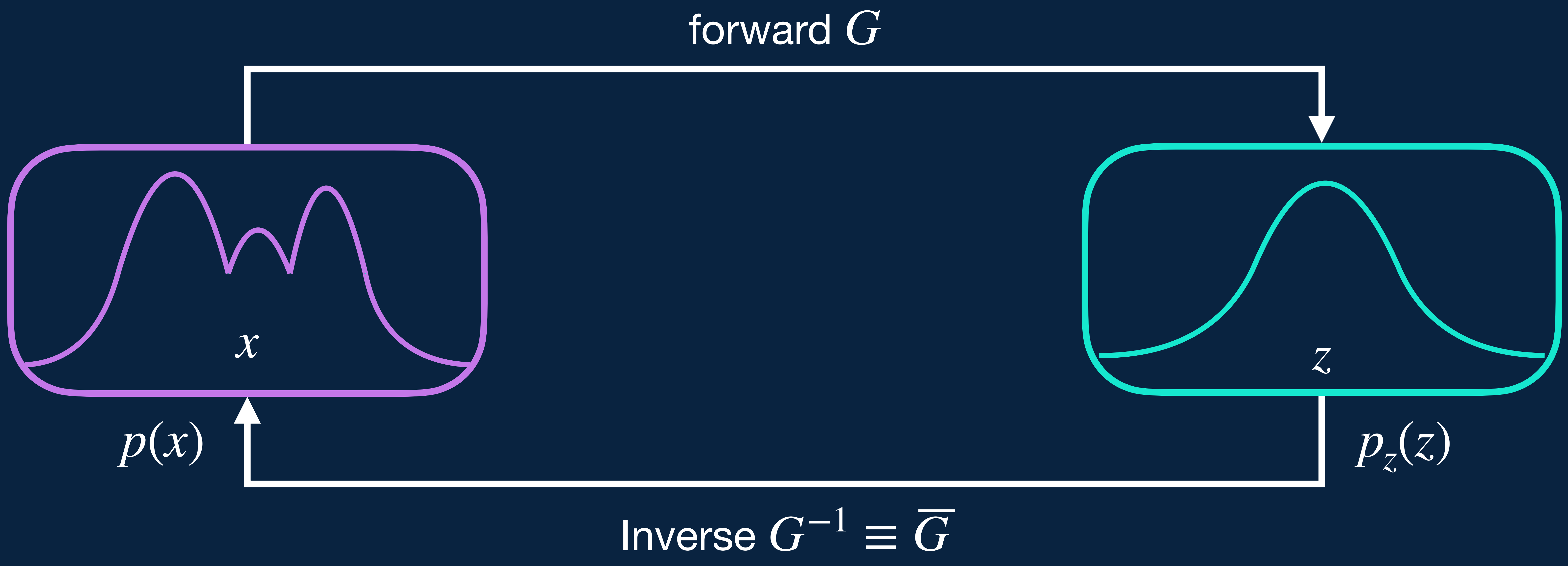
Part II

Normalizing Flows

Normalizing flow — Basics

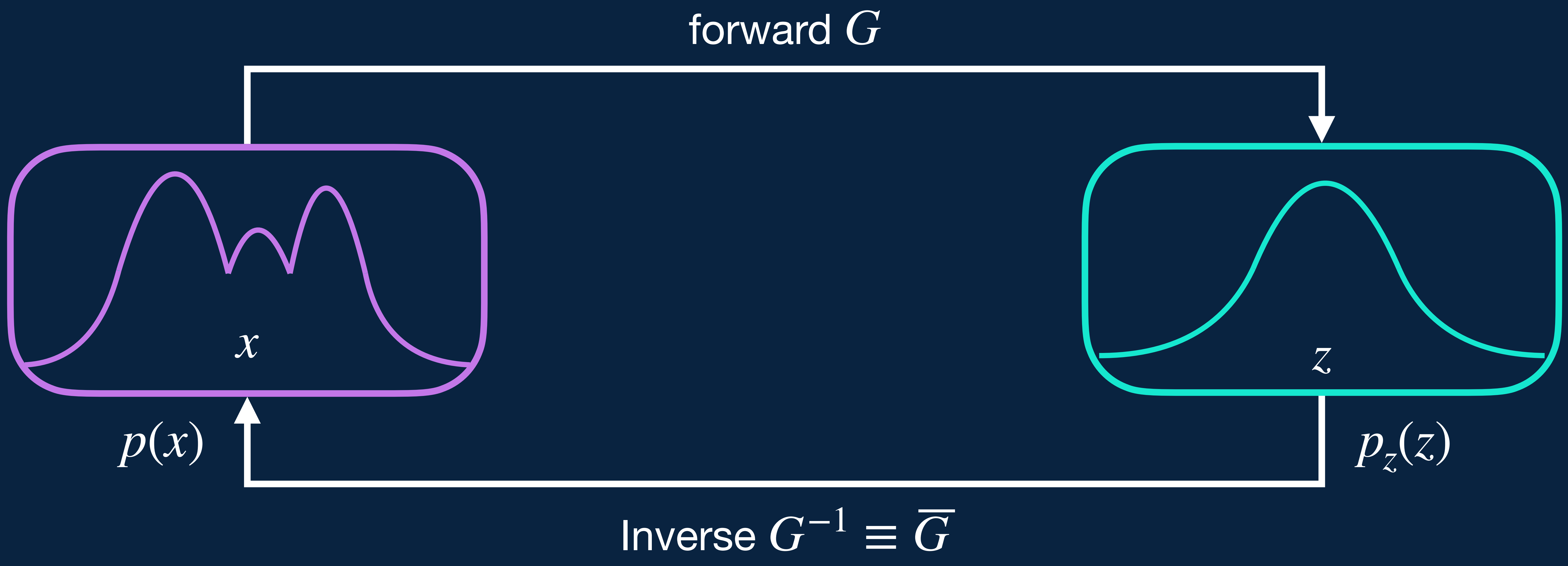


Normalizing flow — Basics



Conservation of probability: $\boxed{p(x)\,\mathrm{d}x = p_z(z)\,\mathrm{d}z}$ with $\boxed{z = G(z) \quad x = \overline{G}(x)}$

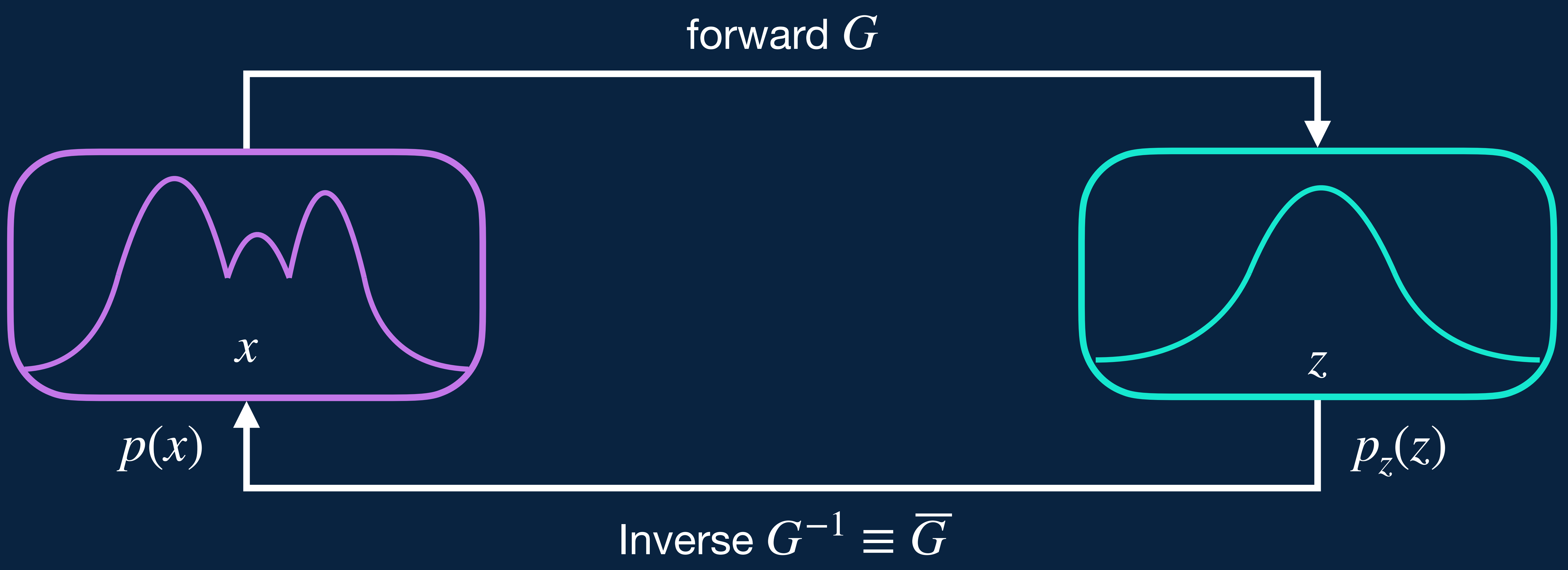
Normalizing flow — Basics



Conservation of probability: $\boxed{p(x)\,\mathrm{d}x = p_z(z)\,\mathrm{d}z}$ with $\boxed{z = G(z) \quad x = \overline{G}(x)}$

Change-of-variables formula: $$\boxed{p_\theta(x) = p_z(z = G_\theta(x)) \cdot \left| \frac{\partial G_\theta(x)}{\partial x} \right|}$$

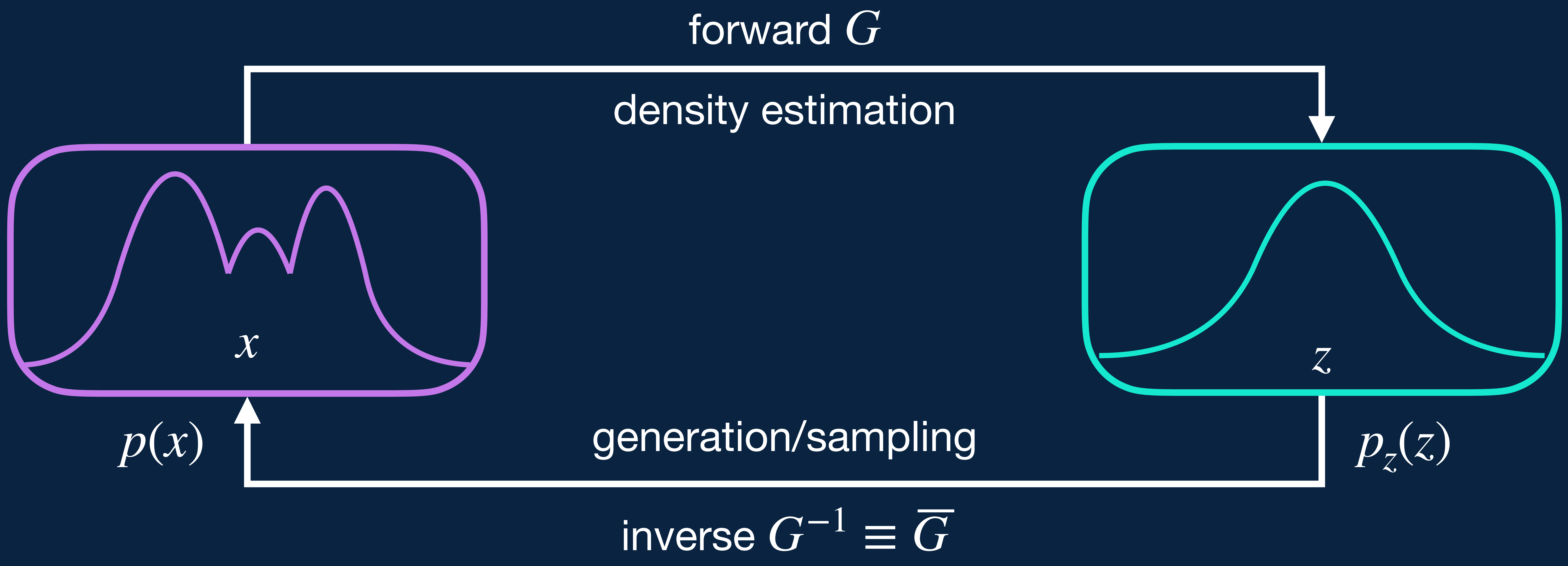
Normalizing flow — Basics



Conservation of probability: $\boxed{p(x)\,\mathrm{d}x = p_z(z)\,\mathrm{d}z}$ with $\boxed{z = G(z) \quad x = \overline{G}(x)}$

Change-of-variables formula:
$$\boxed{\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|}$$

Normalizing flow — Basics

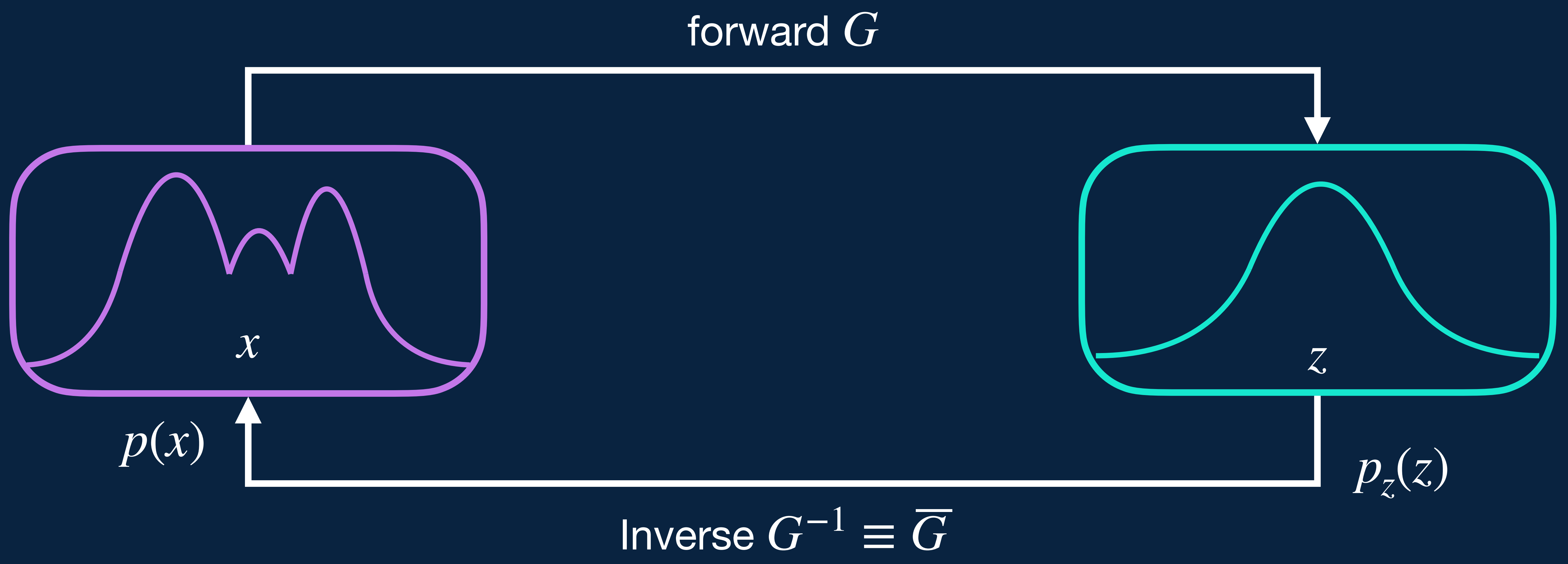


Conservation of probability: $\boxed{p(x)\,\mathrm{d}x = p_z(z)\,\mathrm{d}z}$ with $\boxed{z = G(z) \quad x = \overline{G}(x)}$

Change-of-variables formula: $$\boxed{\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|}$$

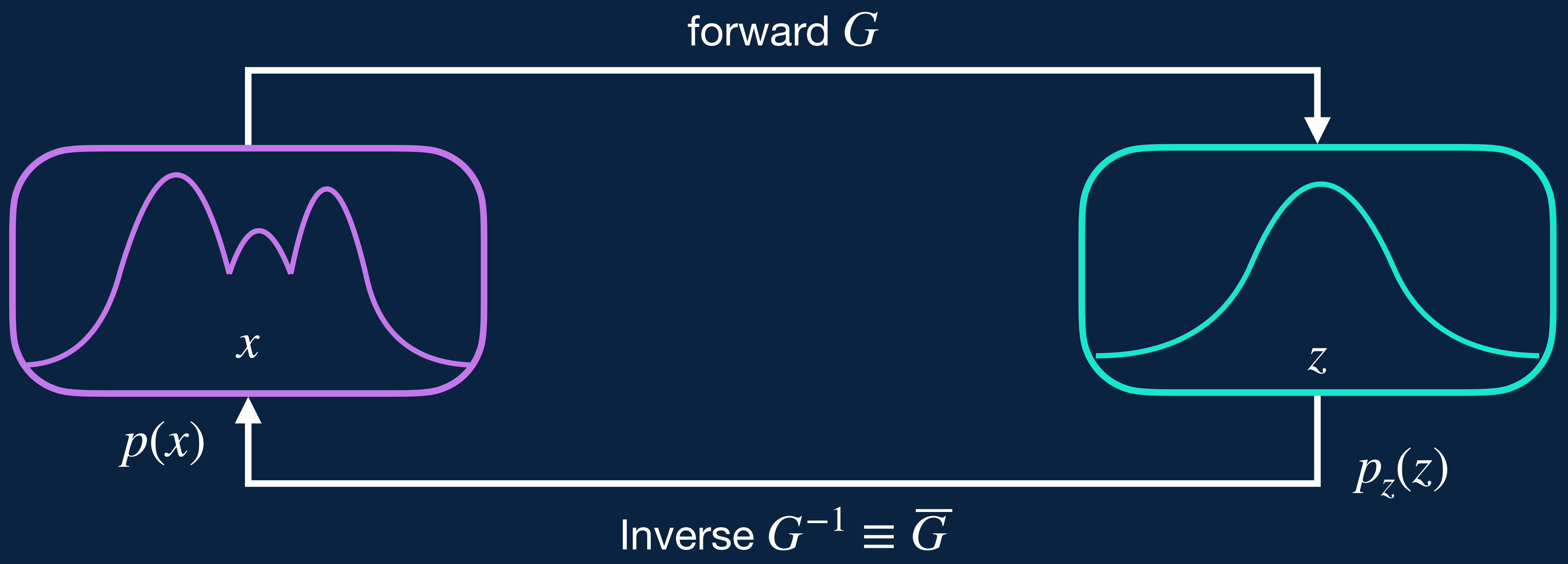
How to train it?

Normalizing flow — Training



$$\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|$$

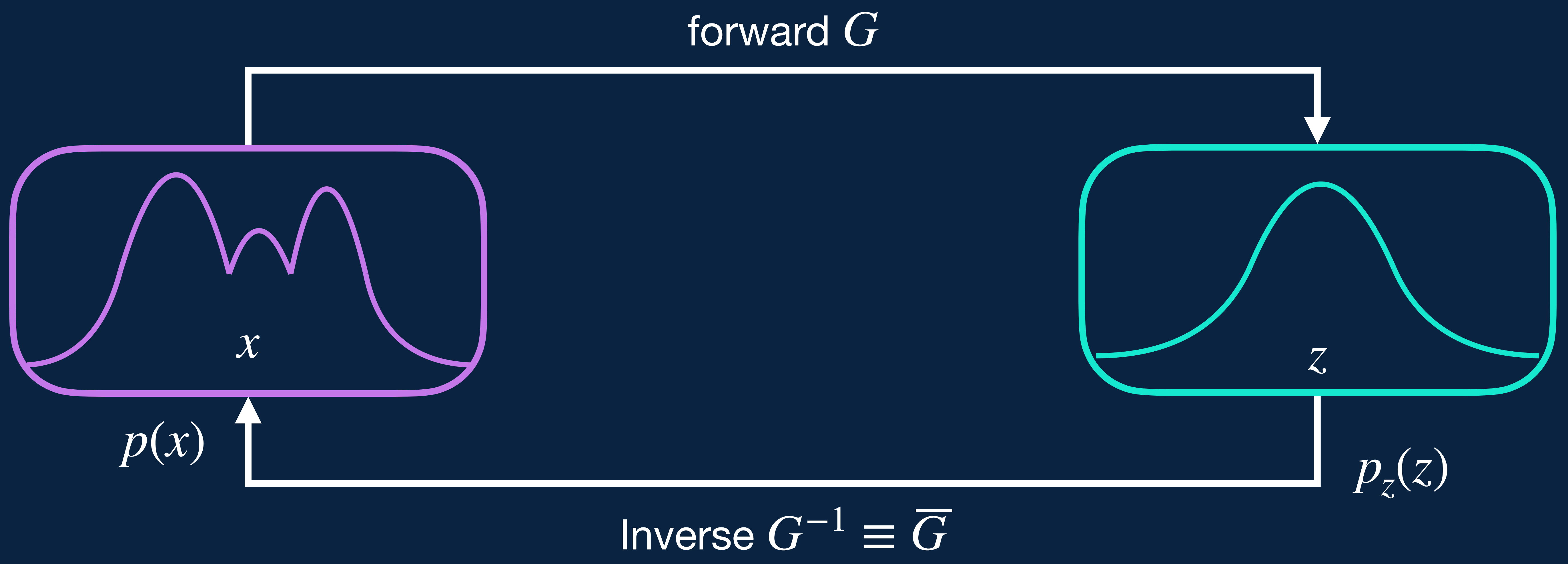
Normalizing flow — Training



$$\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|$$

→ Match $p_\theta(x)$ with $p_{\text{data}}(x)$

Normalizing flow — Training



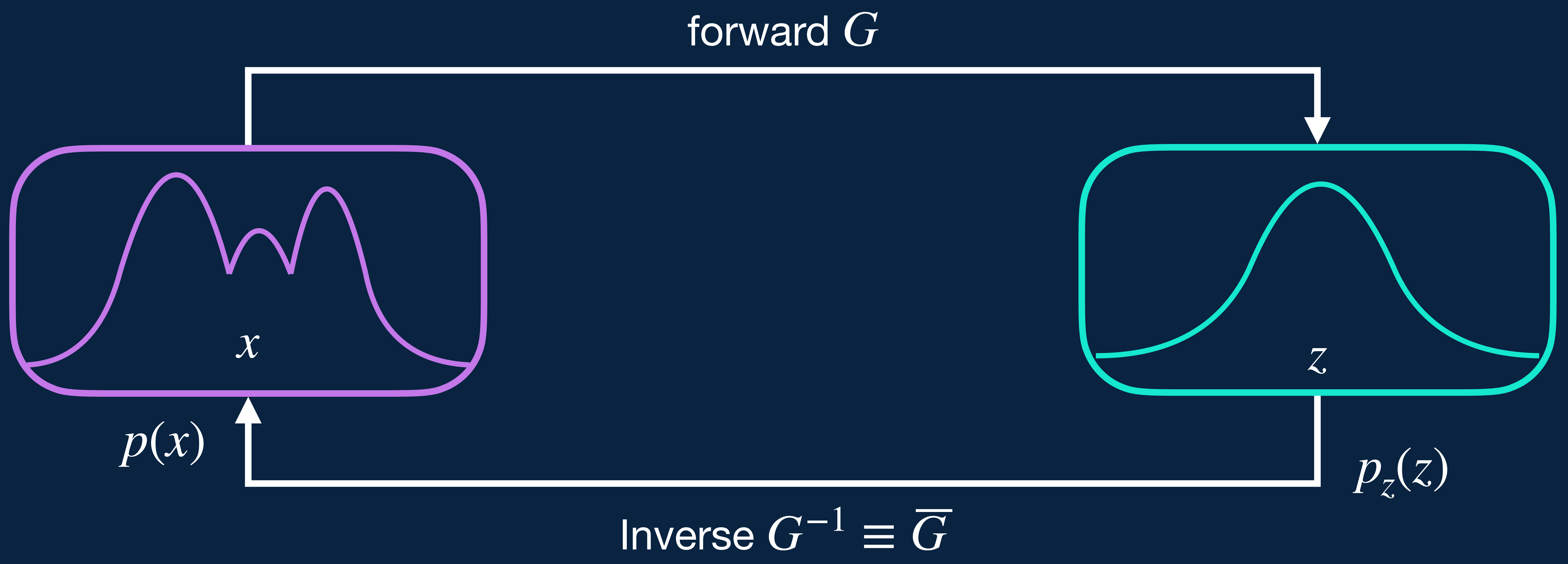
$$\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|$$

→ Match $p_\theta(x)$ with $p_{\mathrm{data}}(x)$

Kullback-Leibler divergence:

$$\begin{aligned} \mathrm{KL}(p_{\mathrm{data}}(x) \,|\, p_\theta(x)) &= \int \mathrm{d}x\, p_{\mathrm{data}}(x)\, \log \frac{p_{\mathrm{data}}(x)}{p_\theta(x)} \\ &= -\int \mathrm{d}x\, p_{\mathrm{data}}(x)\, \log p_\theta(x) + \int \mathrm{d}x\, p_{\mathrm{data}}(x)\, p_{\mathrm{data}}(x) \end{aligned}$$

Normalizing flow — Training



$$\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|$$

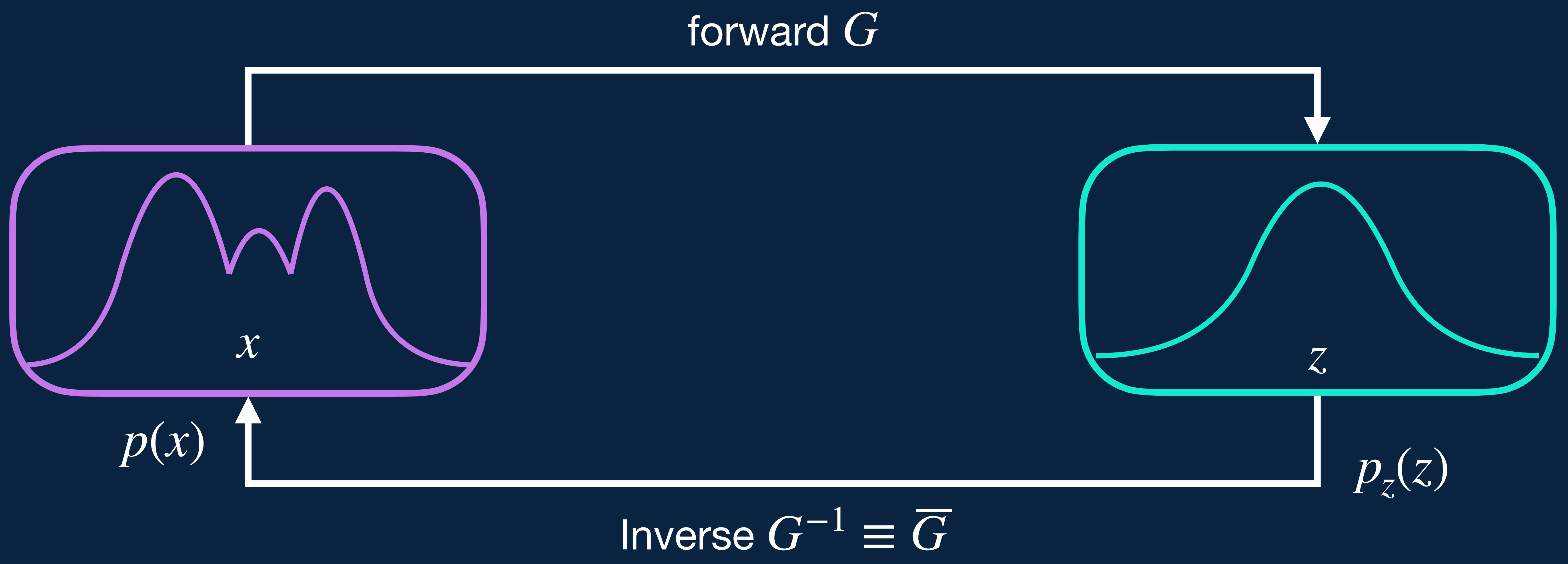
→ Match $p_\theta(x)$ with $p_{\text{data}}(x)$

Kullback-Leibler divergence:

$$\begin{aligned} \mathrm{KL}(p_{\text{data}}(x) \,|\, p_\theta(x)) &= \int \mathrm{d}x\, p_{\text{data}}(x)\, \log \frac{p_{\text{data}}(x)}{p_\theta(x)} \\ &= -\int \mathrm{d}x\, p_{\text{data}}(x)\, \log p_\theta(x) + \boxed{\int \mathrm{d}x\, p_{\text{data}}(x)\, p_{\text{data}}(x)} \end{aligned}$$

No θ dependene

Normalizing flow — Training



$$\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|$$

→ Match $p_\theta(x)$ with $p_{\text{data}}(x)$

Kullback-Leibler divergence:

$$\begin{aligned} \mathrm{KL}(p_{\text{data}}(x) \,|\, p_\theta(x)) &= \int \mathrm{d}x \, p_{\text{data}}(x) \, \log \frac{p_{\text{data}}(x)}{p_\theta(x)} \\ &= -\int \mathrm{d}x \, p_{\text{data}}(x) \, \log p_\theta(x) + \boxed{\int \mathrm{d}x \, p_{\text{data}}(x) \, p_{\text{data}}(x)} \end{aligned}$$

No θ dependene

Negative log-likelihood loss:

$$L_{\text{NLL}} = -\int \mathrm{d}x \, p_{\text{data}}(x) \, \log p_\theta(x) \approx \left\langle -\log p_\theta(x) \right\rangle_{x \sim p_{\text{data}}}$$

Tractable Jacobian?

$$\boxed{\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|} \longrightarrow$$

Requires tractable Jacobian!

Tractable Jacobian?

$$\boxed{\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|}$$

→ Requires tractable Jacobian!

In general: $\boxed{g(x) = \left| \frac{\partial G_\theta(x)}{\partial x} \right| \text{ is } d \times d \text{ matrix}}$ → Scales with $\mathcal{O}(d^3)$ ☹

Tractable Jacobian?

$$\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|$$ → Requires tractable Jacobian!

In general: $g(x) = \left| \frac{\partial G_\theta(x)}{\partial x} \right|$ is $d \times d$ matrix → Scales with $\mathcal{O}(d^3)$ ☹

Solution: **Autoregressive transformations** $z = \begin{pmatrix} z_1 \\ \vdots \\ z_d \end{pmatrix}$ $x = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$

Tractable Jacobian?

$$\boxed{\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|}$$

→ Requires tractable Jacobian!

In general: $\boxed{g(x) = \left| \frac{\partial G_\theta(x)}{\partial x} \right| \text{ is } d \times d \text{ matrix}}$ → Scales with $\mathcal{O}(d^3)$ ☹

Solution: **Autoregressive transformations** $z = \begin{pmatrix} z_1 \\ \vdots \\ z_d \end{pmatrix}$ $x = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$

$$\boxed{\begin{aligned} z_1 &\equiv z_1(x_1) \\ z_2 &\equiv z_2(x_1, x_2) \\ &\vdots \\ z_d &\equiv z_d(x_1, x_2, \ldots, x_d) \end{aligned}}$$

Tractable Jacobian?

$$\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|$$ → Requires tractable Jacobian!

In general: $g(x) = \left| \frac{\partial G_\theta(x)}{\partial x} \right|$ is $d \times d$ matrix → Scales with $\mathcal{O}(d^3)$ ☹

Solution: **Autoregressive transformations** $z = \begin{pmatrix} z_1 \\ \vdots \\ z_d \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$

$$\begin{aligned} z_1 &\equiv z_1(x_1) \\ z_2 &\equiv z_2(x_1, x_2) \\ &\vdots \\ z_d &\equiv z_d(x_1, x_2, \ldots, x_d) \end{aligned}$$

→

$$J_{ij}(x) = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_2}{\partial x_1} & \cdots & \frac{\partial z_d}{\partial x_1} \\ 0 & \frac{\partial z_2}{\partial x_2} & \cdots & \frac{\partial z_d}{\partial x_1} \\ \vdots & \ddots & & \vdots \\ 0 & \cdots & 0 & \frac{\partial z_d}{\partial x_d} \end{pmatrix}$$

Tractable Jacobian?

$$\log p_\theta(x) = \log p_z(z = G_\theta(x)) + \log \left| \frac{\partial G_\theta(x)}{\partial x} \right|$$ → Requires tractable Jacobian!

In general: $g(x) = \left| \frac{\partial G_\theta(x)}{\partial x} \right|$ is $d \times d$ matrix → Scales with $\mathcal{O}(d^3)$ ☹

Solution: **Autoregressive transformations** $z = \begin{pmatrix} z_1 \\ \vdots \\ z_d \end{pmatrix}$ $x = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix}$

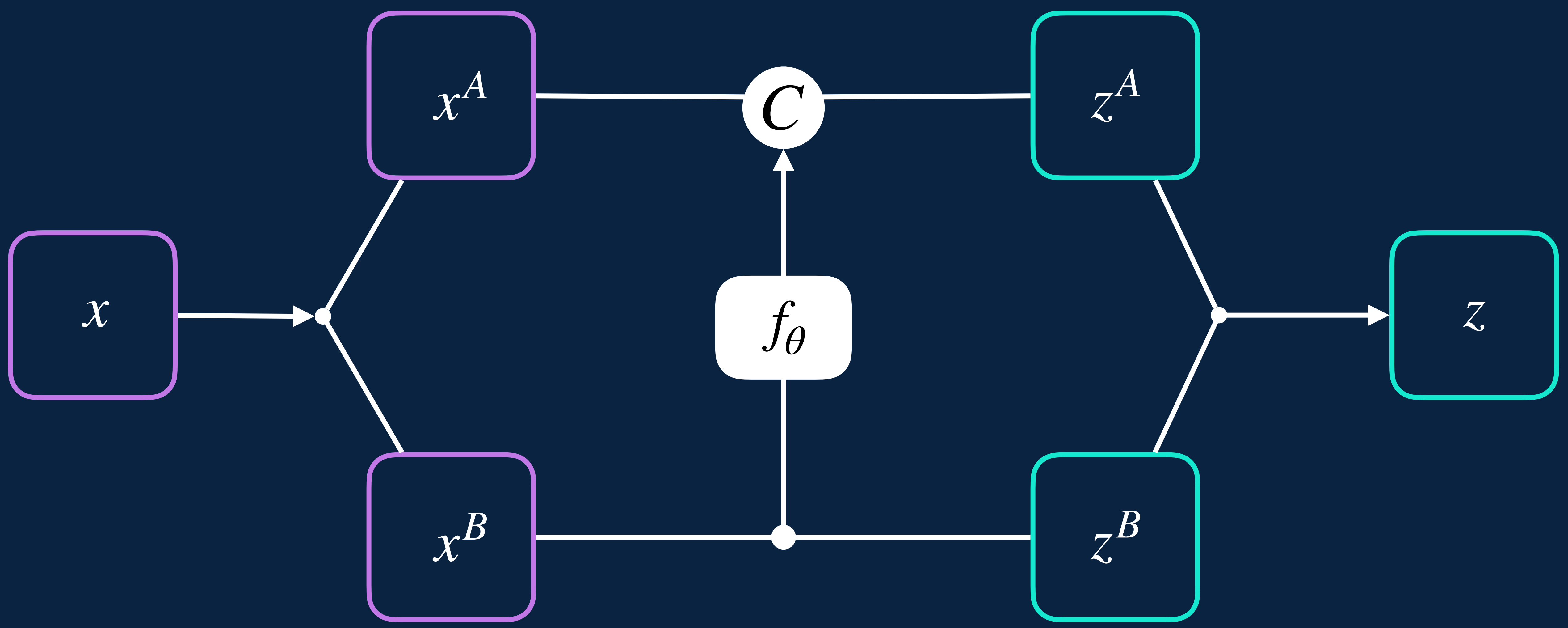
$$\begin{aligned} z_1 &\equiv z_1(x_1) \\ z_2 &\equiv z_2(x_1, x_2) \\ &\vdots \\ z_d &\equiv z_d(x_1, x_2, \ldots, x_d) \end{aligned}$$

→

$$J_{ij}(x) = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_2}{\partial x_1} & \cdots & \frac{\partial z_d}{\partial x_1} \\ 0 & \frac{\partial z_2}{\partial x_2} & \cdots & \frac{\partial z_d}{\partial x_1} \\ \vdots & \ddots & & \vdots \\ 0 & \cdots & 0 & \frac{\partial z_d}{\partial x_d} \end{pmatrix}$$

→ $\det J = \prod_i J_{ii} \sim \mathcal{O}(d)$ ☺

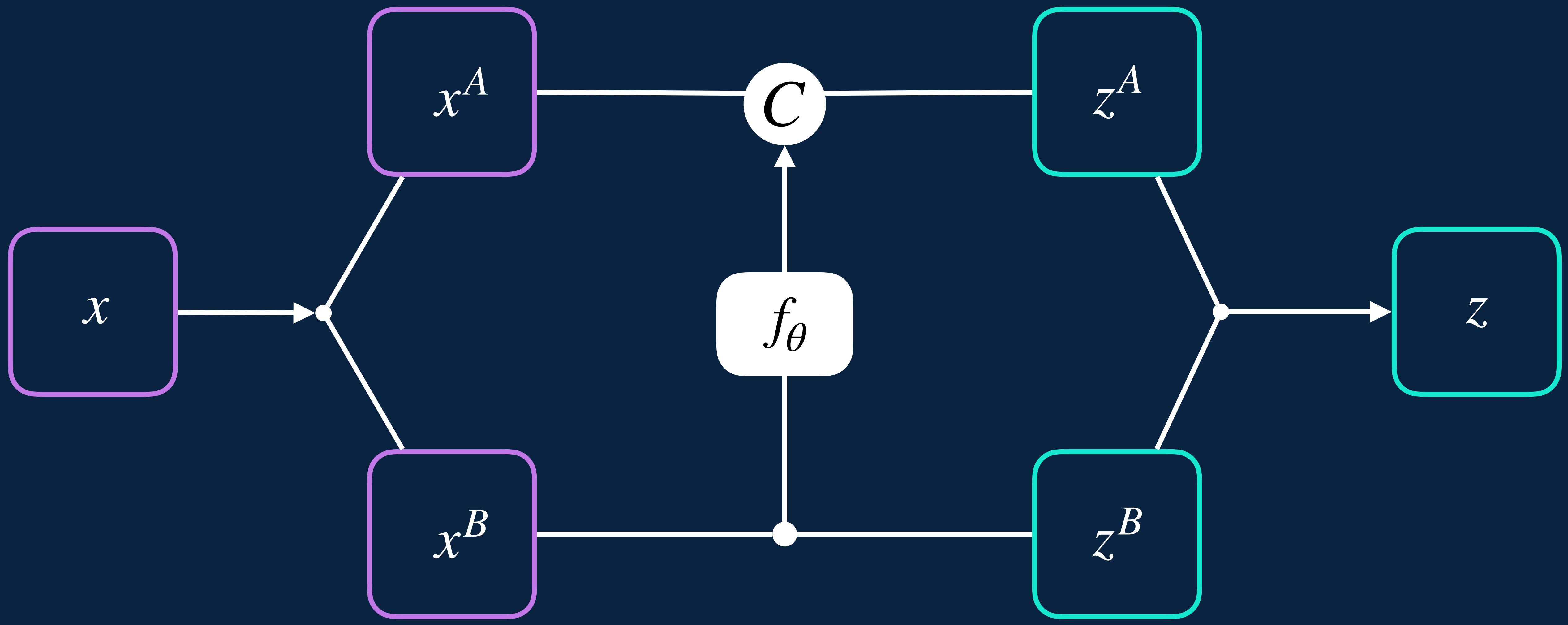
Coupling block



Forward pass:

$$z^A = C(x^A; f_\theta(x^B))$$
$$z^B = x^B$$

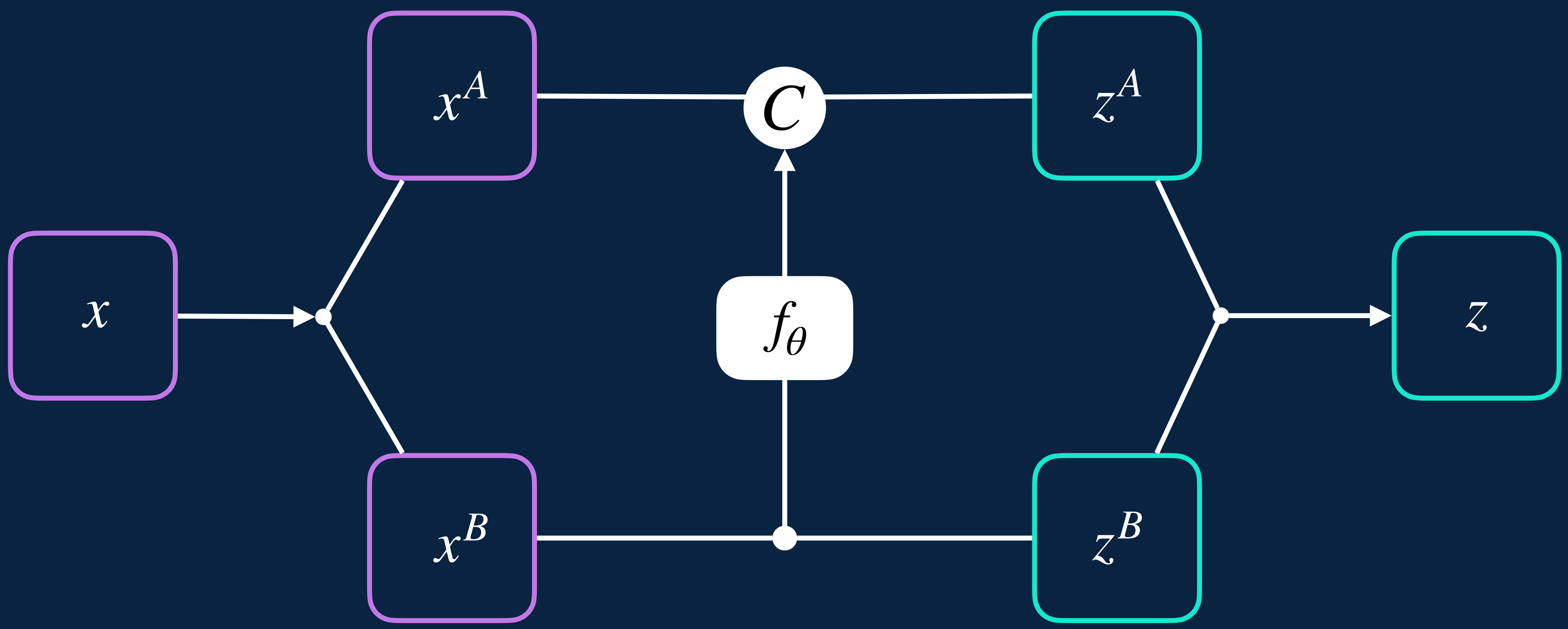
Coupling block



Forward pass:
$$\boxed{\begin{aligned} z^A &= C(x^A; f_\theta(x^B)) \\ z^B &= x^B \end{aligned}}$$

$$\boxed{J_{ij}(x) = \begin{pmatrix} \frac{\partial C}{\partial x^A} & \frac{\partial C}{\partial f_\theta}\frac{\partial f_\theta}{\partial x^B} \\ 0 & I_m \end{pmatrix}}$$

Coupling block



Forward pass:

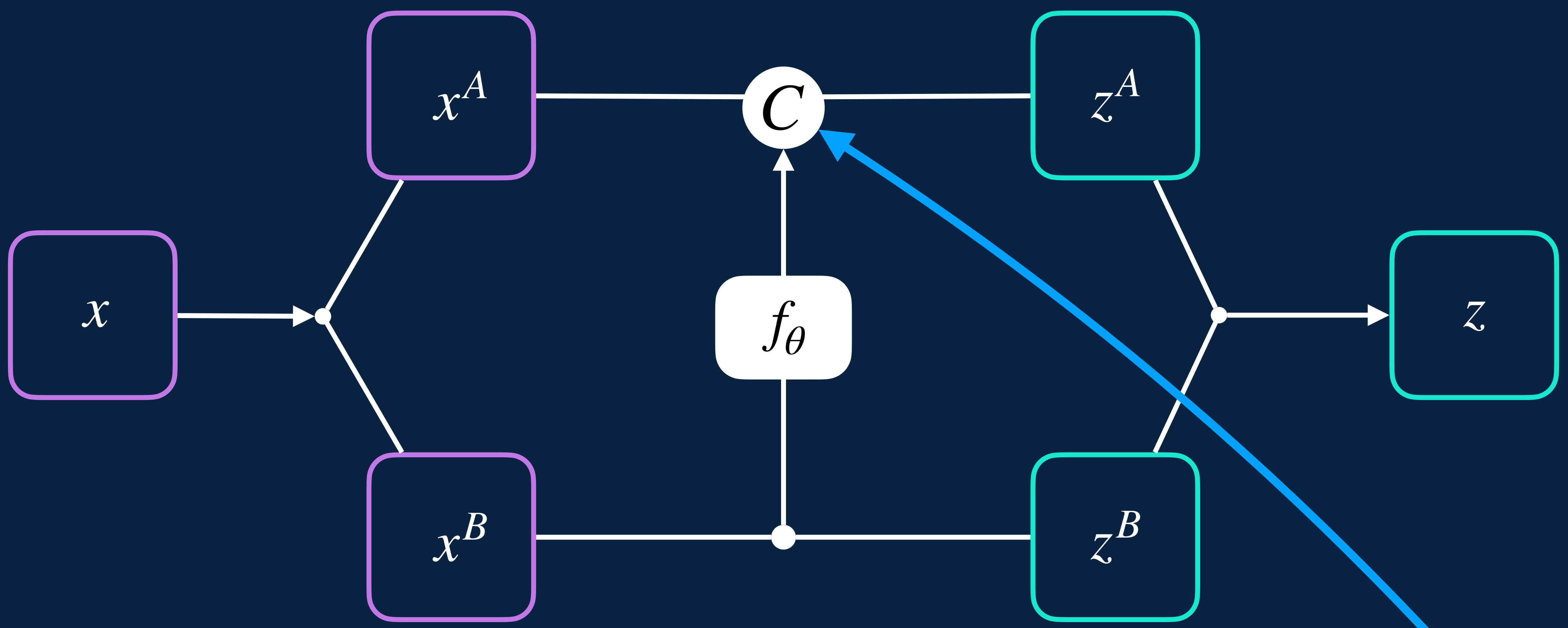
$$z^A = C(x^A; f_\theta(x^B))$$
$$z^B = x^B$$

Inverse pass:

$$x^A = C^{-1}(z^A; f_\theta(z^B))$$
$$x^B = z^B$$

$$J_{ij}(x) = \begin{pmatrix} \frac{\partial C}{\partial x^A} & \frac{\partial C}{\partial f_\theta}\frac{\partial f_\theta}{\partial x^B} \\ 0 & I_m \end{pmatrix}$$

Coupling block



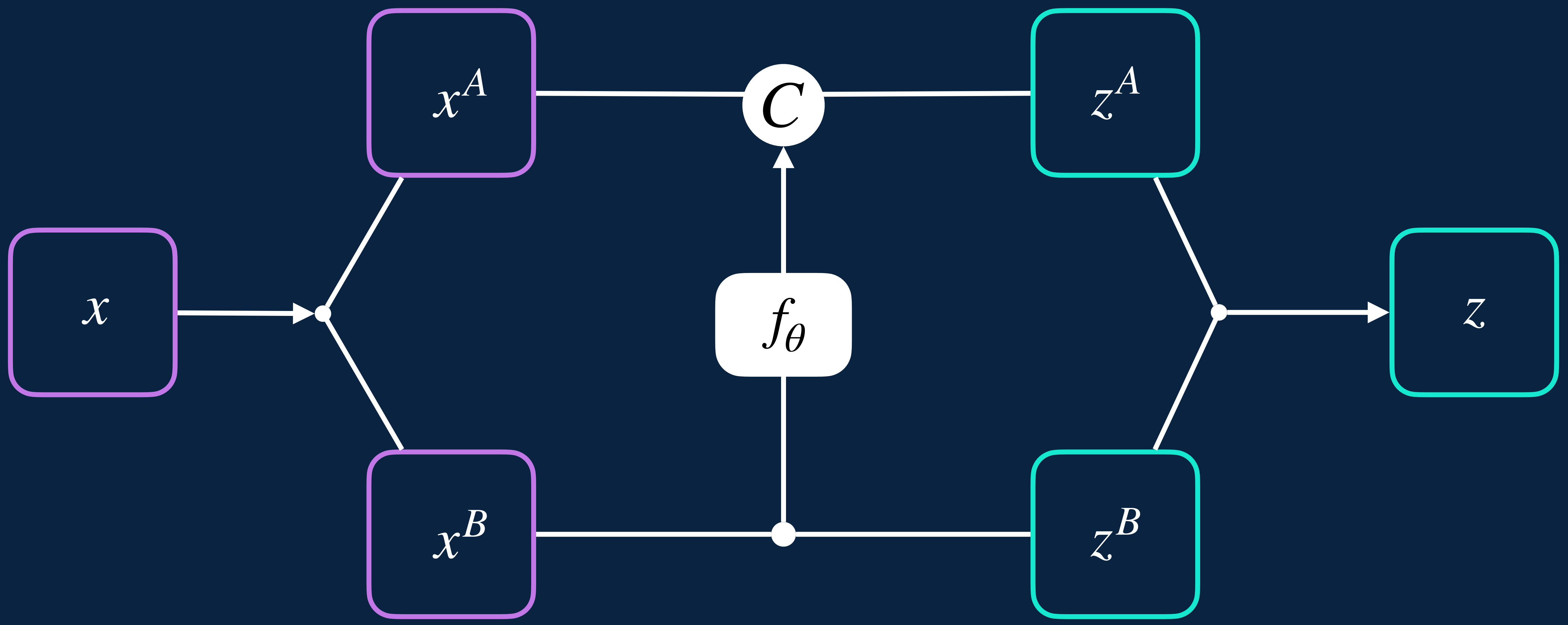
Forward pass:
$$z^A = C(x^A; f_\theta(x^B))$$
$$z^B = x^B$$

Inverse pass:
$$x^A = C^{-1}(z^A; f_\theta(z^B))$$
$$x^B = z^B$$

$$J_{ij}(x) = \begin{pmatrix} \frac{\partial C}{\partial x^A} & \frac{\partial C}{\partial f_\theta}\frac{\partial f_\theta}{\partial x^B} \\ 0 & I_m \end{pmatrix}$$

What is the function C?

Coupling block



Forward pass:
$$z^A = C(x^A; f_\theta(x^B))$$
$$z^B = x^B$$

Inverse pass:
$$x^A = C^{-1}(z^A; f_\theta(z^B))$$
$$x^B = z^B$$

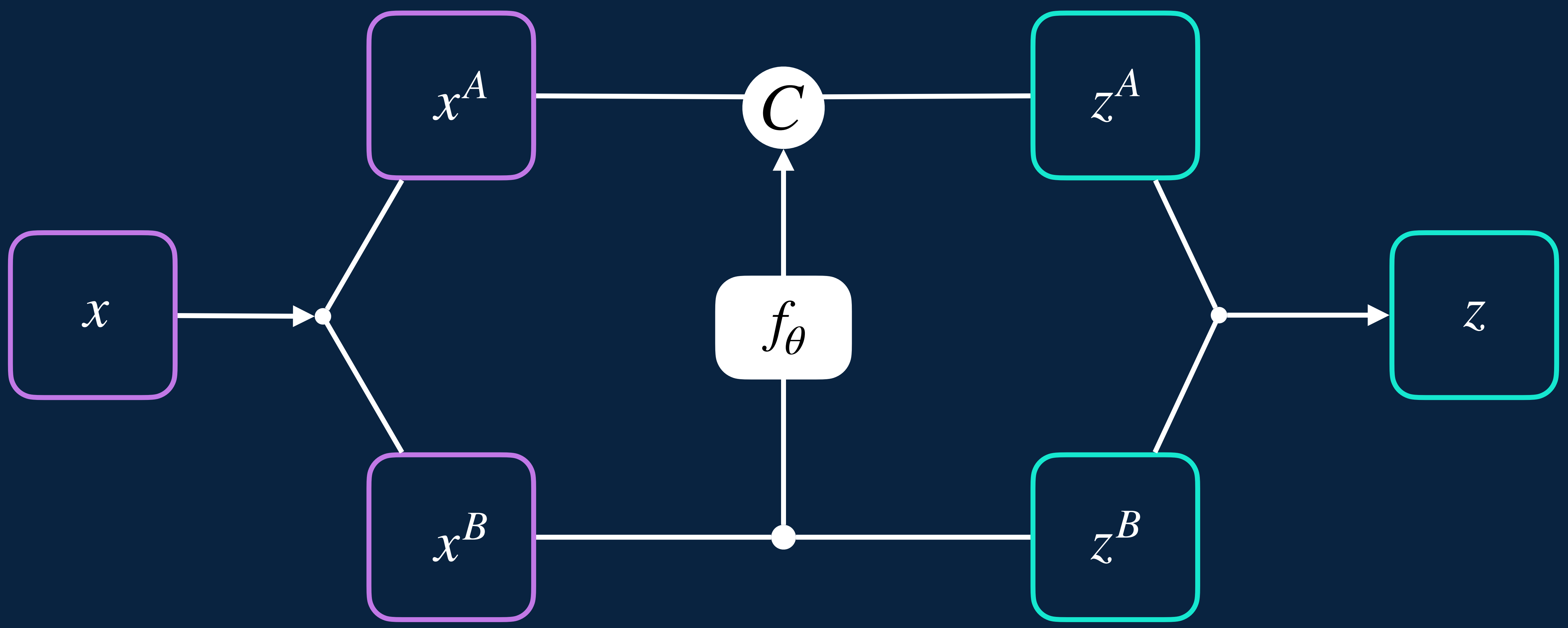
$$J_{ij}(x) = \begin{pmatrix} \frac{\partial C}{\partial x^A} & \frac{\partial C}{\partial f_\theta}\frac{\partial f_\theta}{\partial x^B} \\ 0 & I_m \end{pmatrix}$$

Affine $C^A = \alpha_\theta(x^B) \cdot x^A + \mu_\theta(x^B)$

[1605.08803]

parametrized by NN

Coupling block



Forward pass:

$$z^A = C(x^A; f_\theta(x^B))$$
$$z^B = x^B$$

Inverse pass:

$$x^A = C^{-1}(z^A; f_\theta(z^B))$$
$$x^B = z^B$$

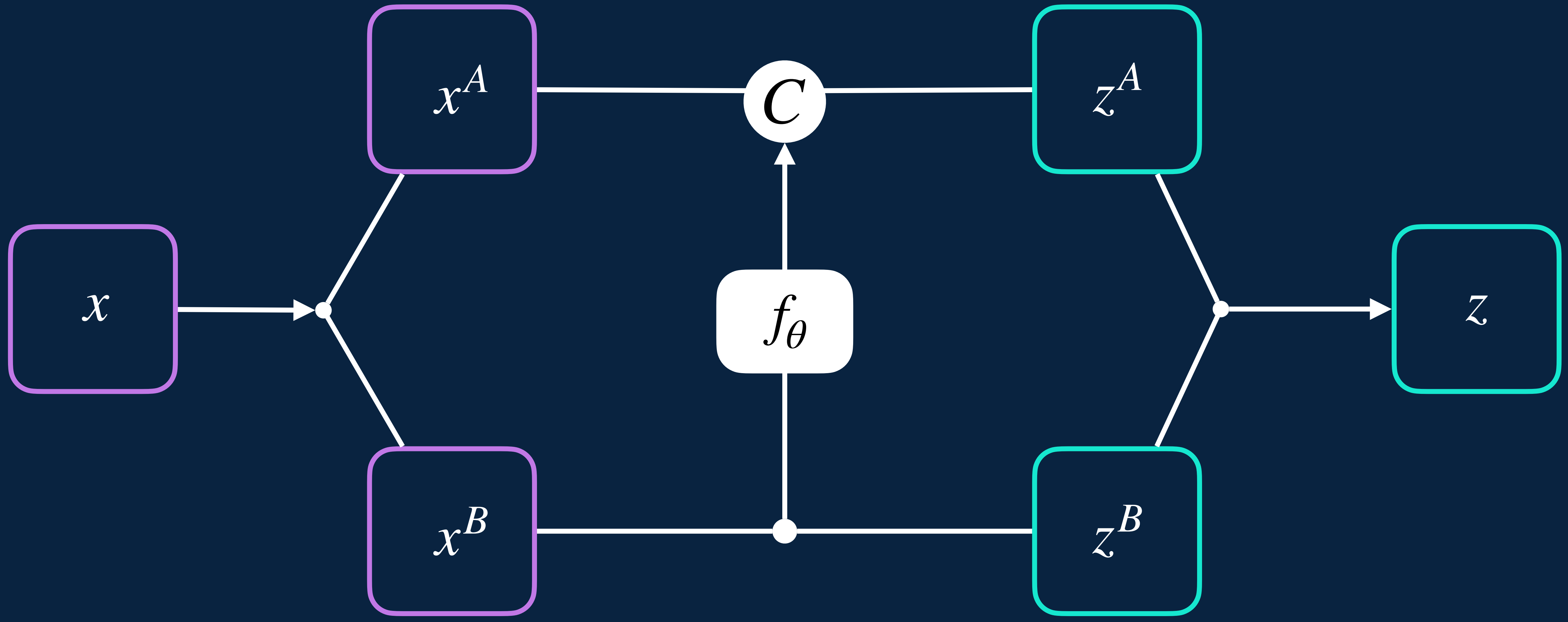
$$J_{ij}(x) = \begin{pmatrix} \frac{\partial C}{\partial x^A} & \frac{\partial C}{\partial f_\theta}\frac{\partial f_\theta}{\partial x^B} \\ 0 & I_m \end{pmatrix}$$

Affine [1605.08803] $C^A = \alpha_\theta(x^B) \cdot x^A + \mu_\theta(x^B)$

Quadratic [1808.03856] $C = a\,x^2 + b\,x + c$

Rational quadratic [1906.04032] $C = \dfrac{a\,x^2 + b\,x + c}{d\,x^2 + e\,x + f}$

Coupling block



Forward pass:

$$z^A = C(x^A; f_\theta(x^B))$$
$$z^B = x^B$$

Inverse pass:

$$x^A = C^{-1}(z^A; f_\theta(z^B))$$
$$x^B = z^B$$

$$J_{ij}(x) = \begin{pmatrix} \frac{\partial C}{\partial x^A} & \frac{\partial C}{\partial f_\theta}\frac{\partial f_\theta}{\partial x^B} \\ 0 & I_m \end{pmatrix}$$

Affine [1605.08803] $C^A = \alpha_\theta(x^B) \cdot x^A + \mu_\theta(x^B)$

Quadratic [1808.03856] $C = a\,x^2 + b\,x + c$

Rational quadratic [1906.04032] $C = \dfrac{a\,x^2 + b\,x + c}{d\,x^2 + e\,x + f}$

Part III — MadNIS

Neural Importance Sampling

[2212.06172]

[2001.05486, 2001.05478, 2001.10028, 2005.12719, 2112.09145]

MadNIS — Basic functionality

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

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

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

MadNIS — Basic functionality

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

$\downarrow$

Use physics knowledge to construct channel and mappings

$\downarrow$ $\downarrow$

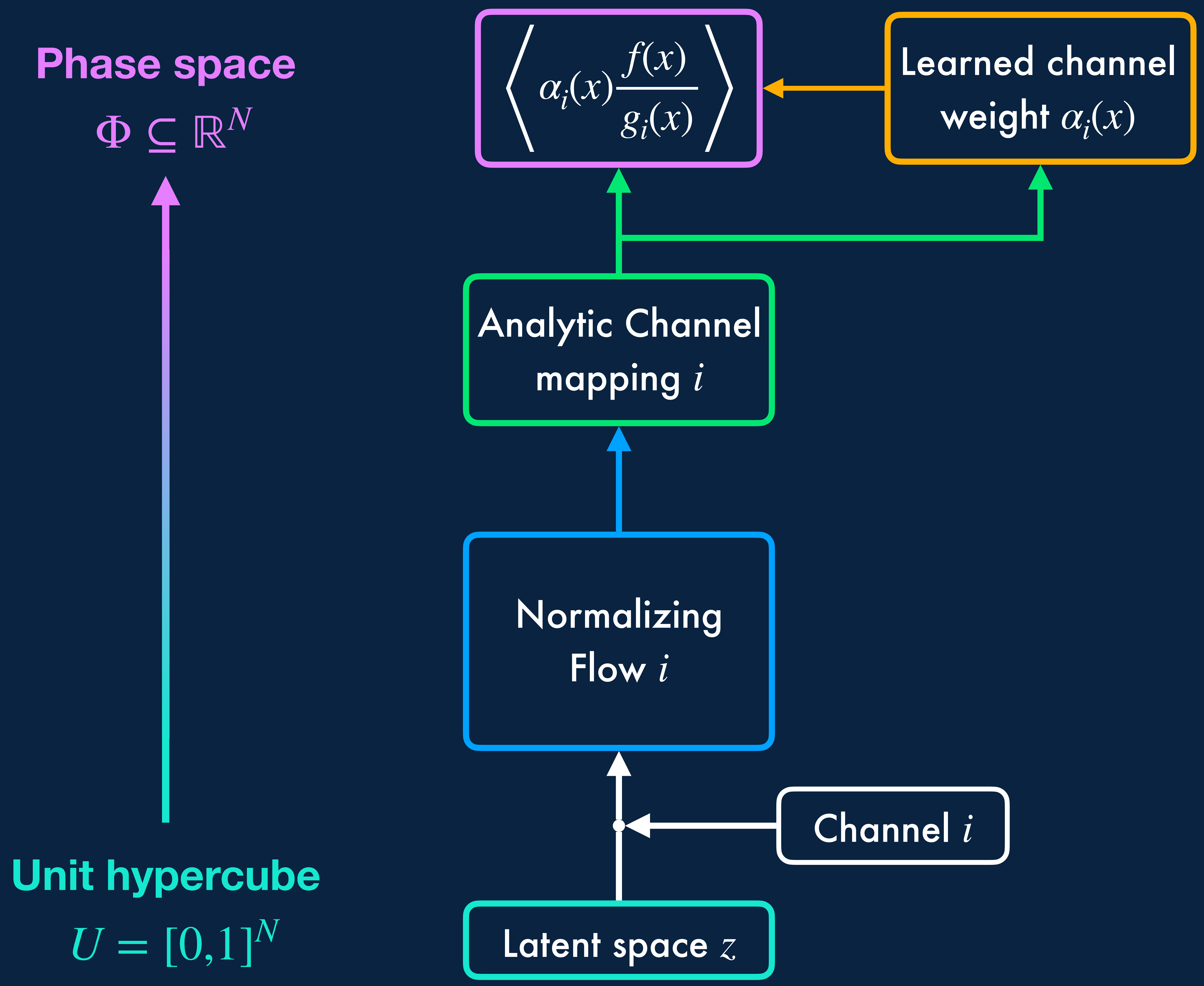
Normalizing flow to refine channel mappings

Fully connected network to refine channel weights

$\downarrow$ $\downarrow$

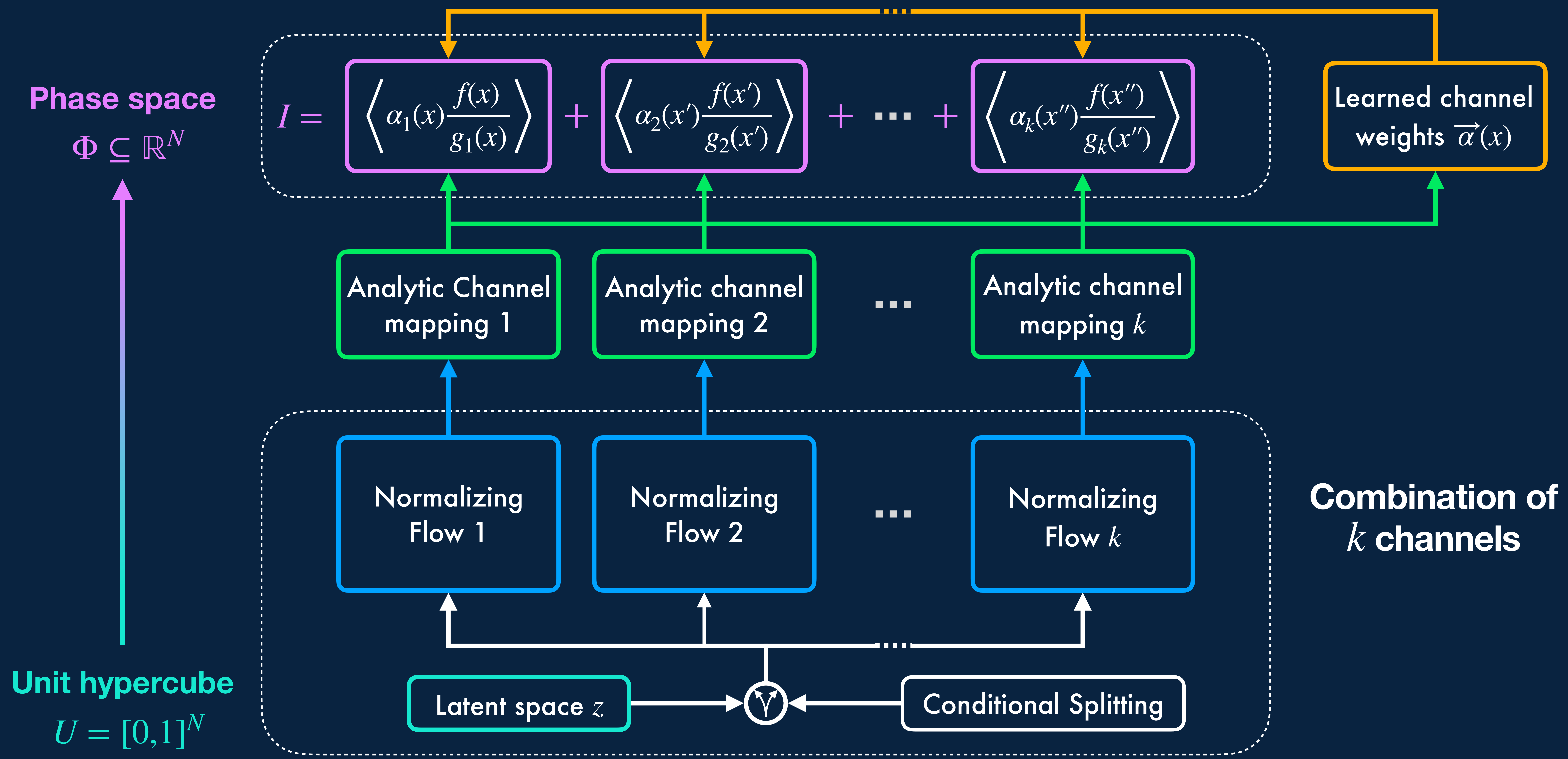
Update simultanously with variance as loss function

MadNIS — Basic functionality



Single channel i

MadNIS — Basic functionality



MadNIS — Overview

MadNIS

Basic functionality

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

Improved multi-channeling

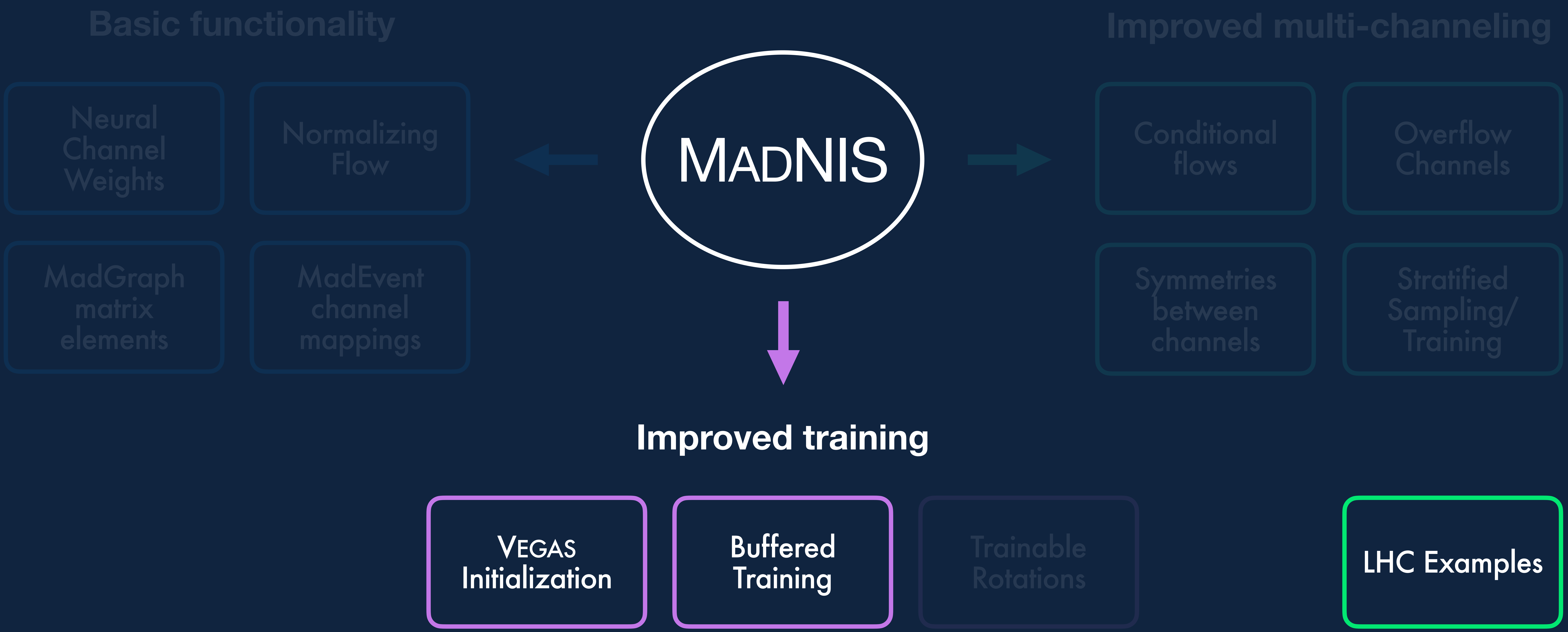
- Conditional flows
- Overflow Channels
- Symmetries between channels
- Stratified Sampling/ Training

Improved training

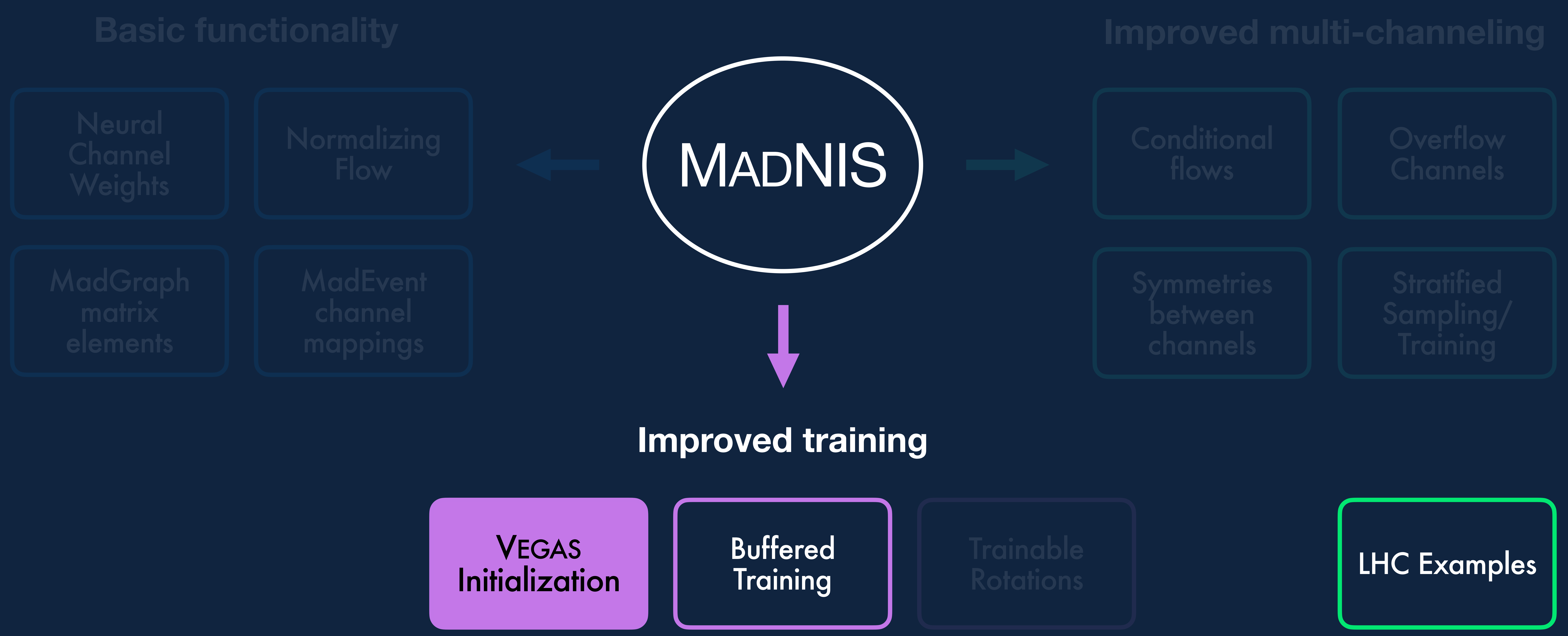
- VEGAS Initialization
- Buffered Training
- Trainable Rotations

LHC Examples

MadNIS — Overview



VEGAS initialization



VEGAS initialization

	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

Combine advantages:

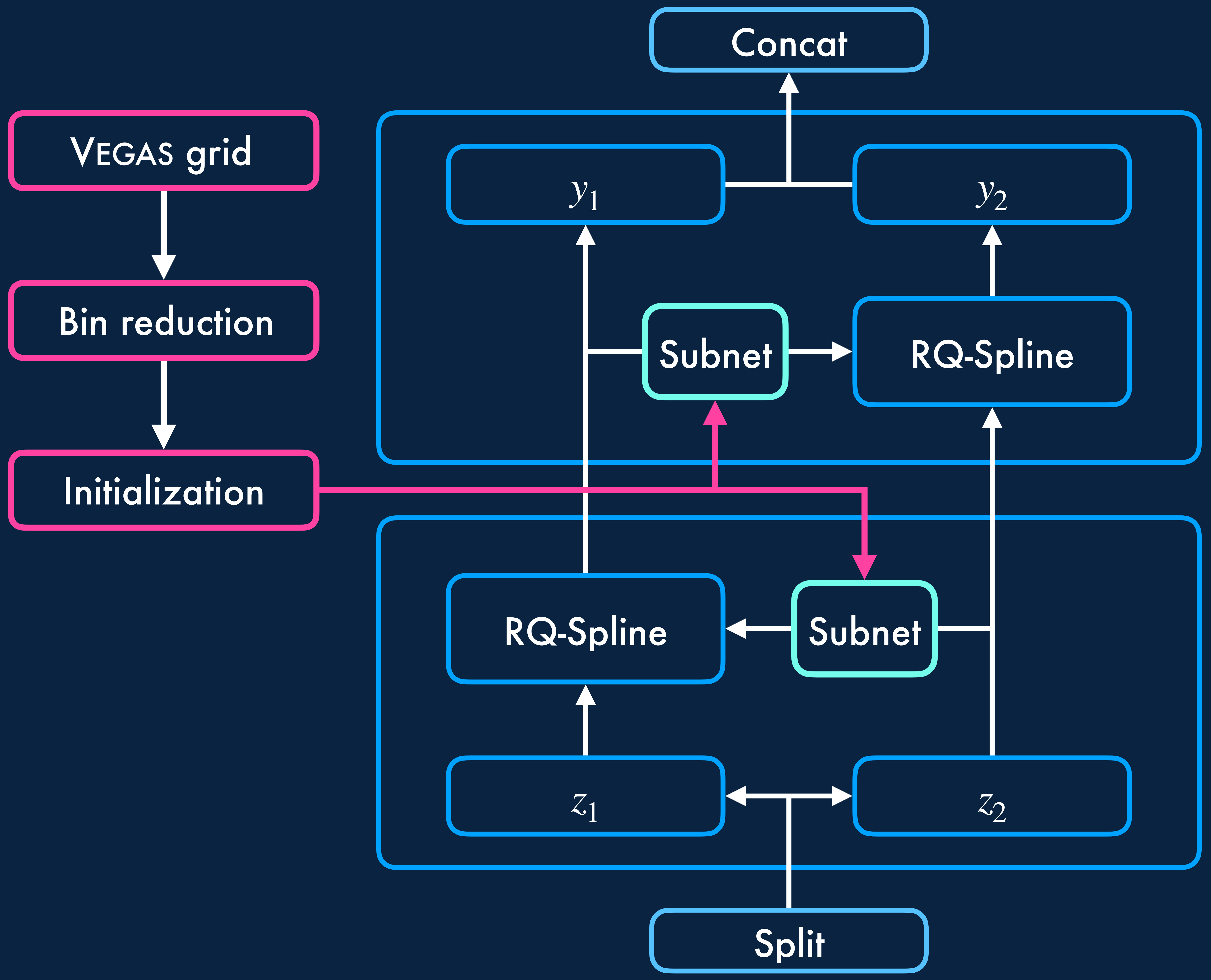
Pre-trained VEGAS grid as starting point for flow training

VEGAS initialization

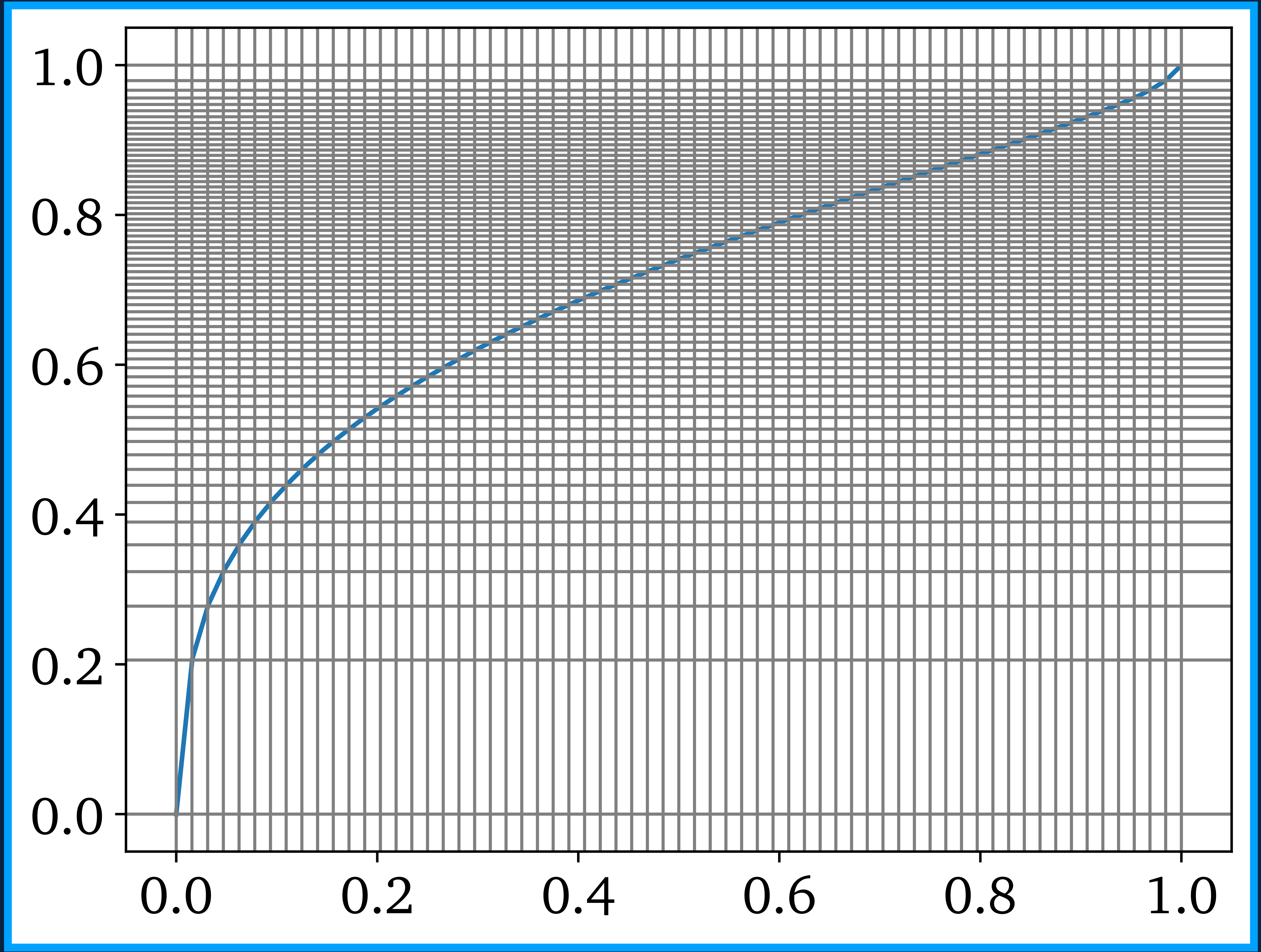
	VEGAS	Flow
Training	Fast	Slow
Correlations	No	Yes

Combine advantages:

Pre-trained VEGAS grid as starting point for flow training

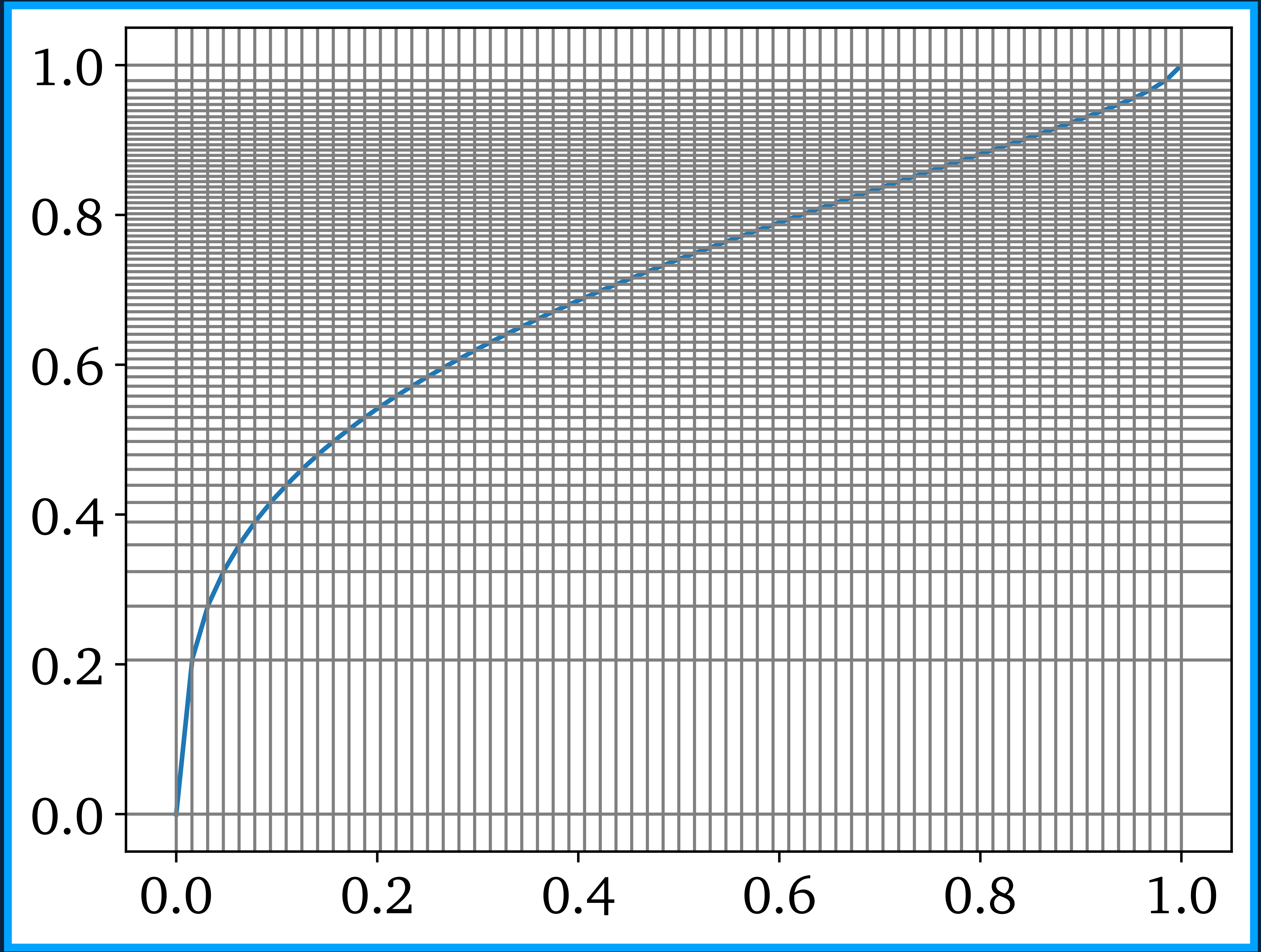


Bin reduction

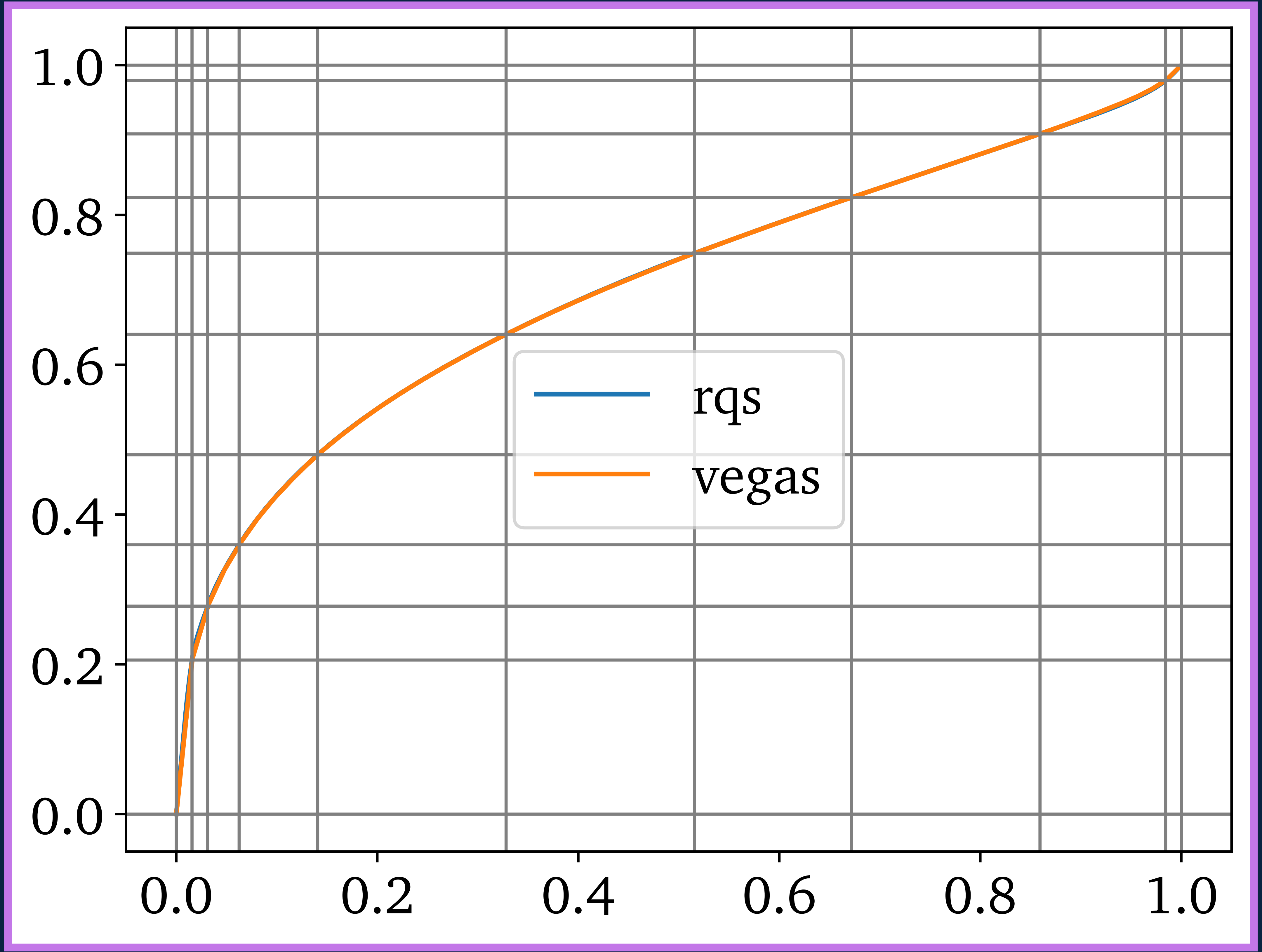


64 VEGAS bins

Bin reduction



64 VEGAS bins



10 RQS bins

Buffered training

Basic functionality

Neural Channel Weights

Normalizing Flow

MadGraph matrix elements

MadEvent channel mappings

MadNIS

Improved multi-channeling

Conditional flows

Overflow Channels

Symmetries between channels

Stratified Sampling/ Training

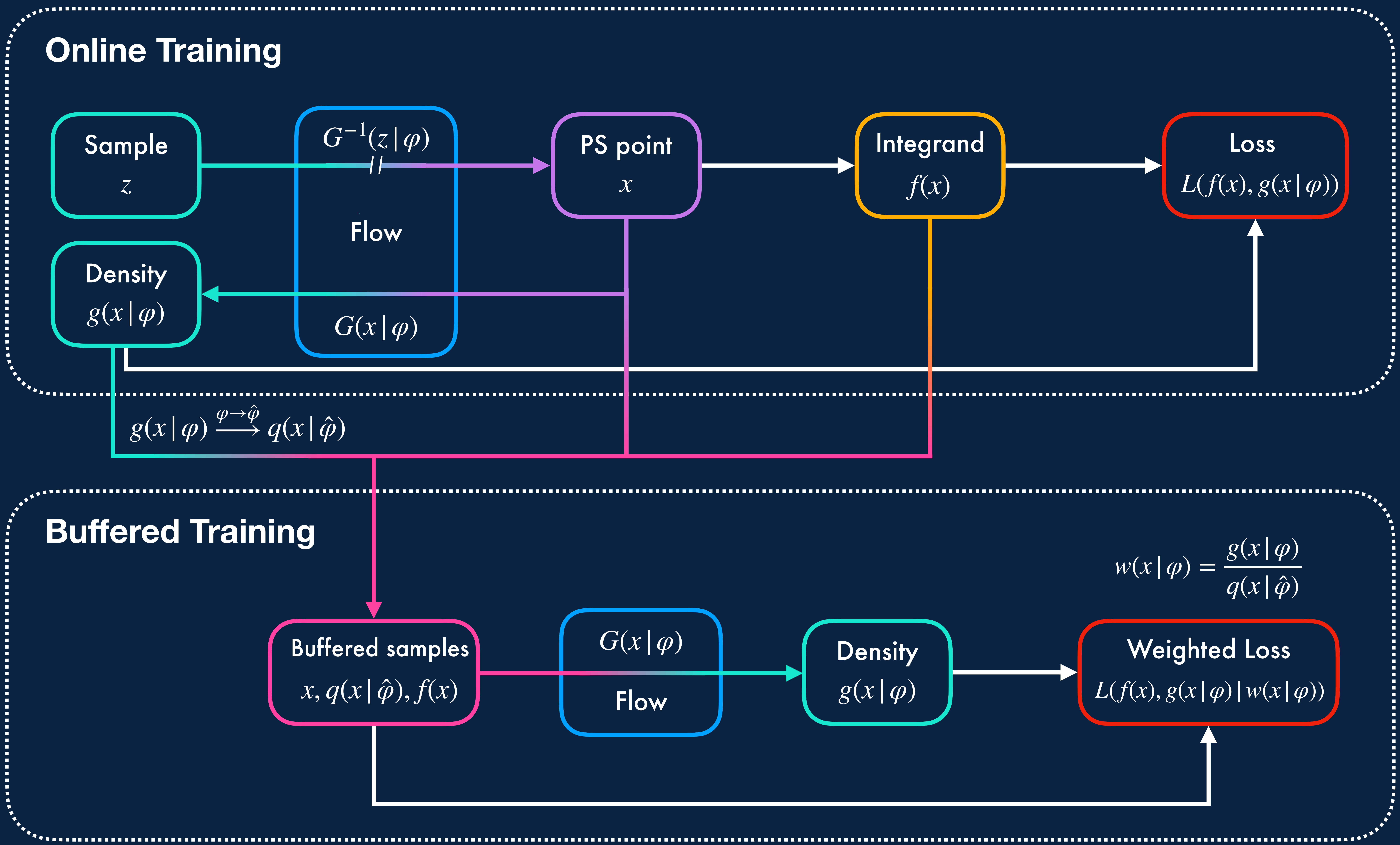
Improved training

VEGAS Initialization

Buffered Training

Trainable Rotations

LHC Examples



Online Training
Sample
z
$G^{-1}(z|\varphi)$
Flow
$G(x|\varphi)$
PS point
x
Integrand
$f(x)$
Loss
$L(f(x), g(x|\varphi))$
Density
$g(x|\varphi)$
$g(x|\varphi) \xrightarrow{\varphi\to\hat{\varphi}} q(x|\hat{\varphi})$
Buffered Training
$w(x|\varphi) = \frac{g(x|\varphi)}{q(x|\hat{\varphi})}$
Buffered samples
$x, q(x|\hat{\varphi}), f(x)$
$G(x|\varphi)$
Flow
Density
$g(x|\varphi)$
Weighted Loss
$L(f(x), g(x|\varphi)|w(x|\varphi))$

Buffered training

Training algorithm

generate new samples, train on them, save samples

↓

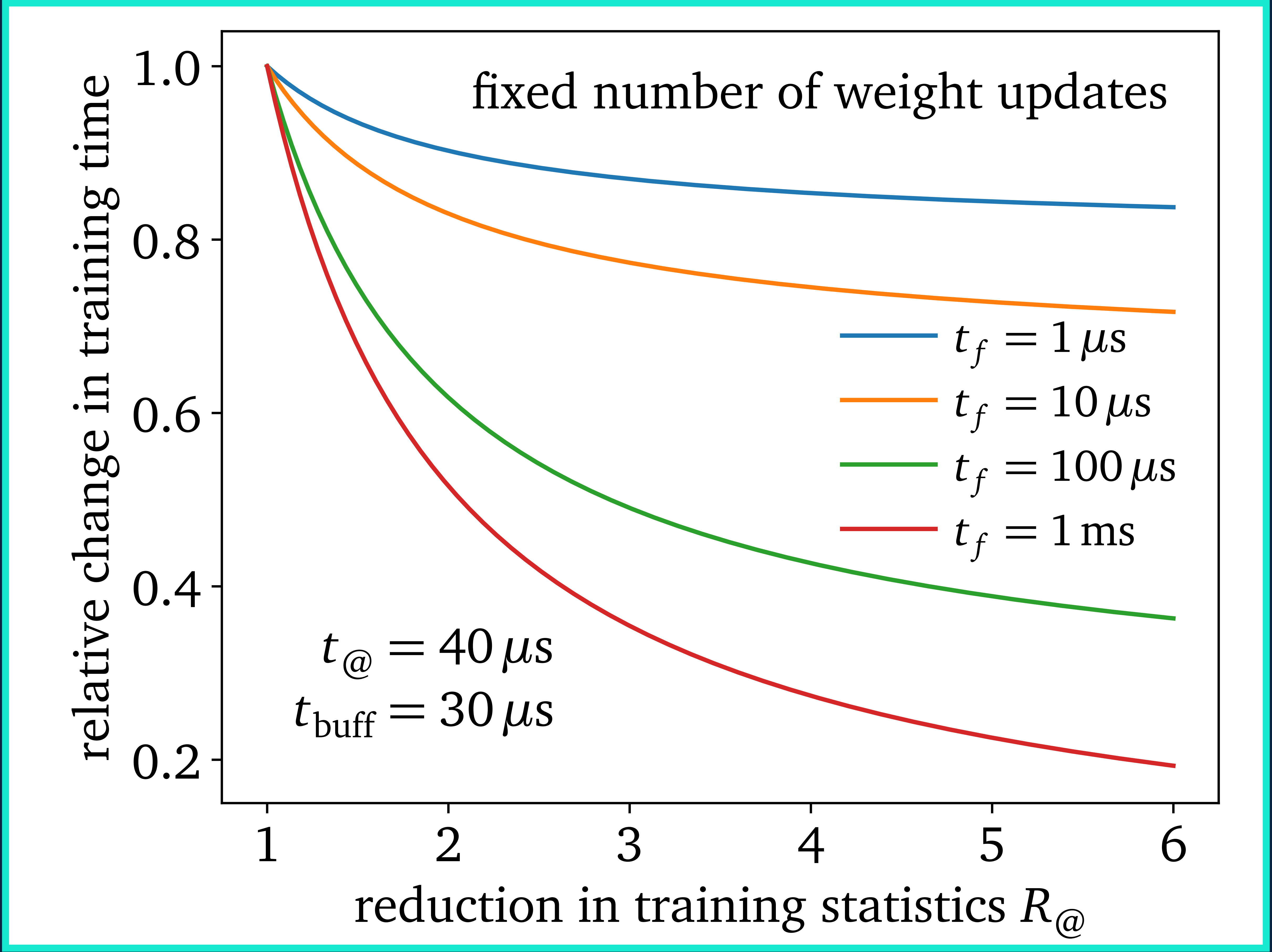
train on saved samples n times

↓

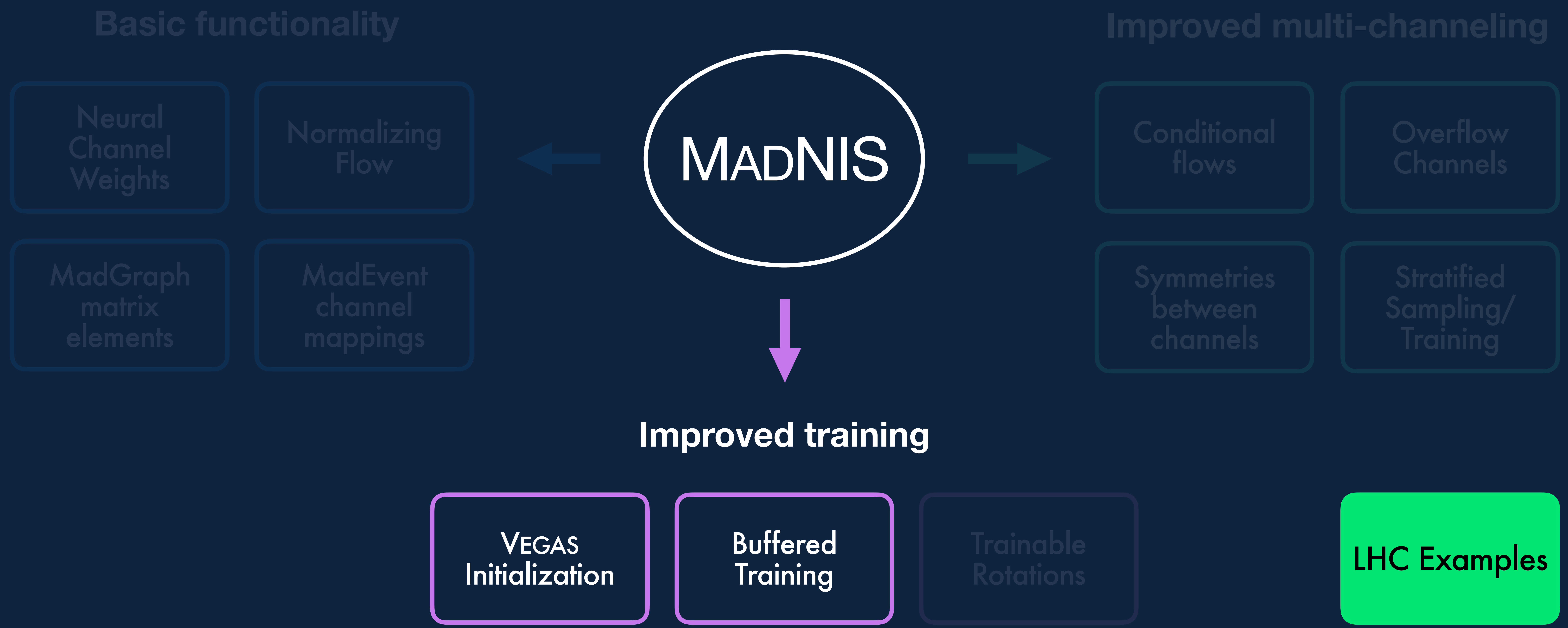
repeat

Reduction in training statistics by

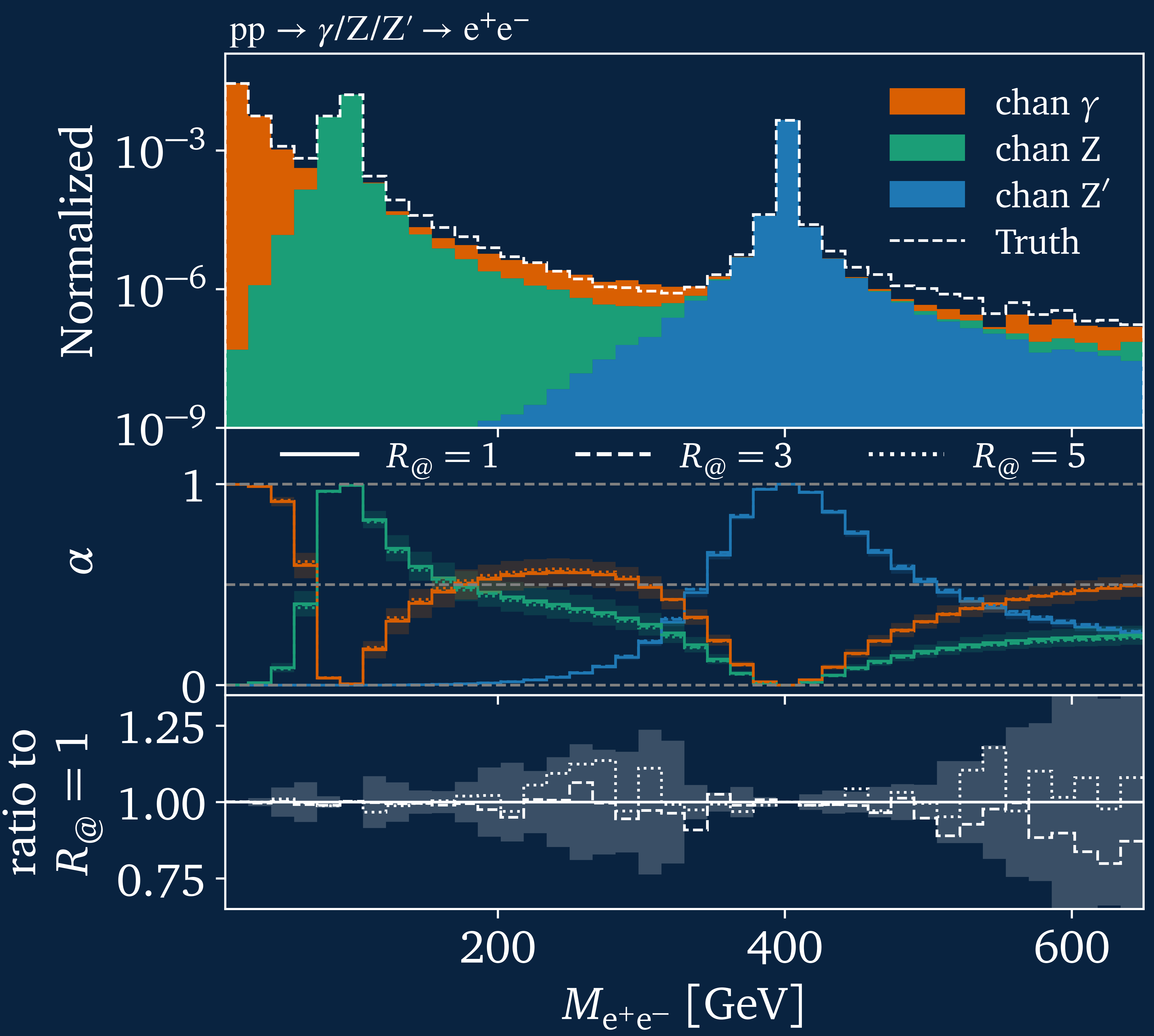
$$R_@ = n + 1$$



LHC examples

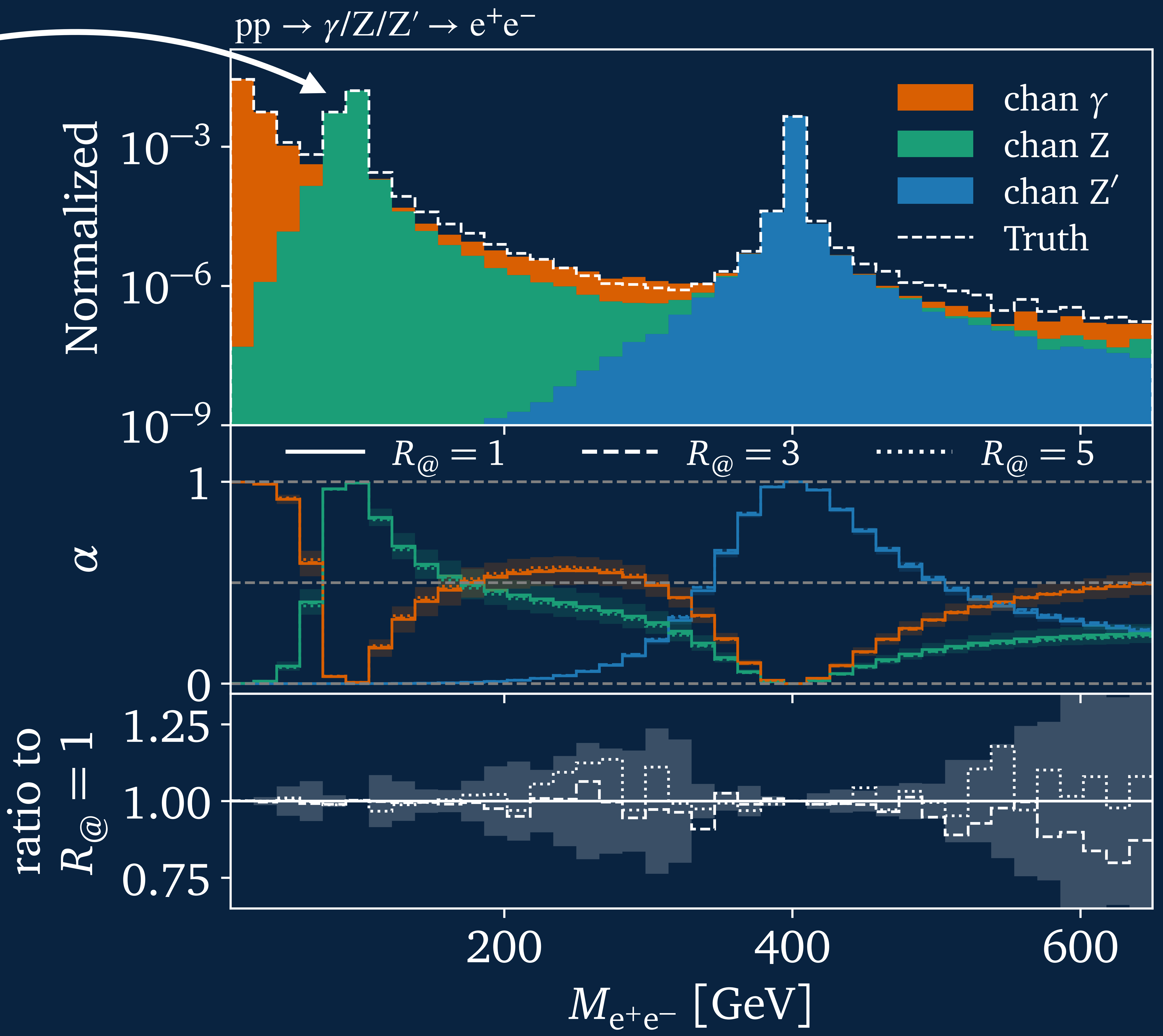


LHC example I — Drell-Yan



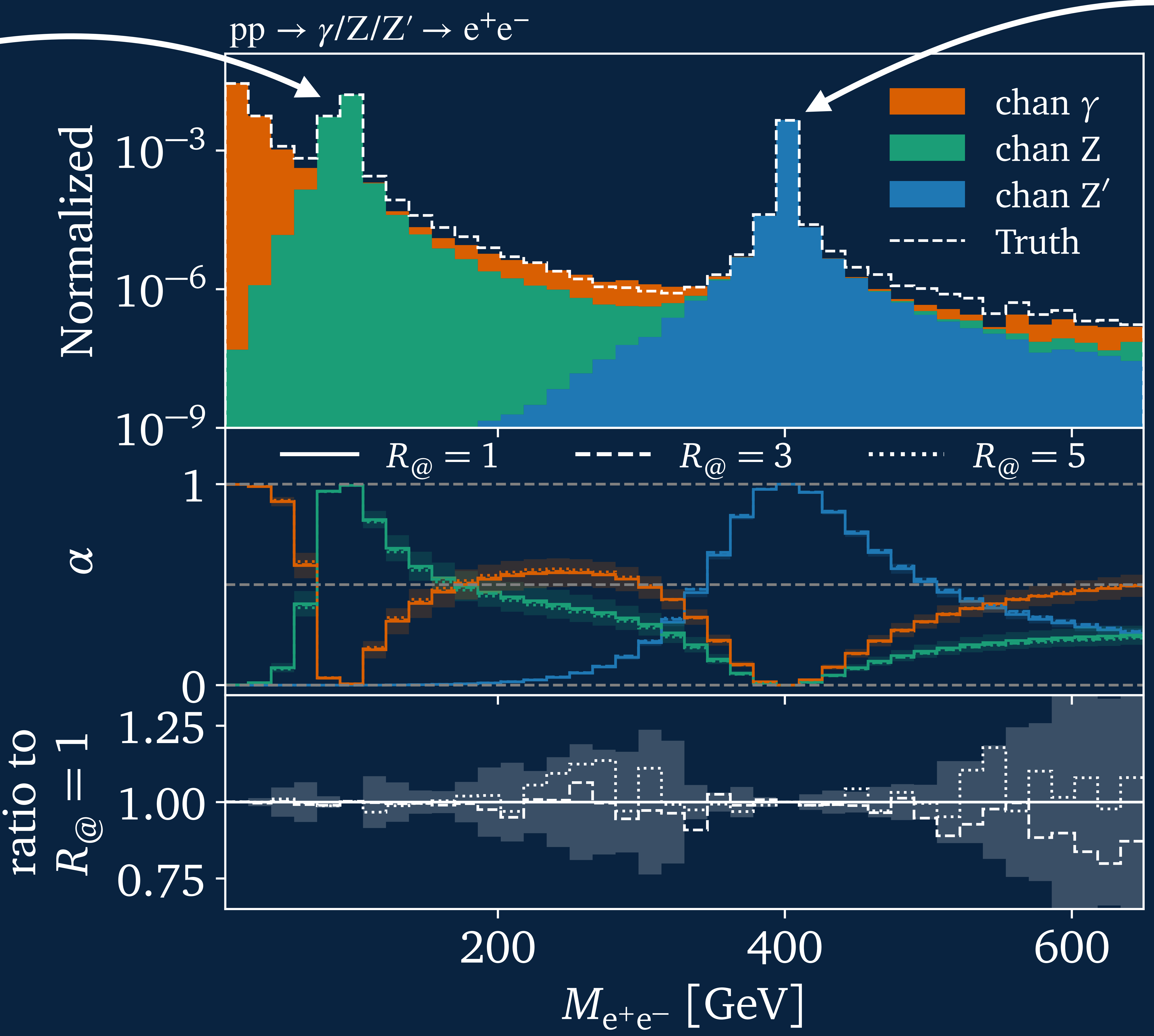
LHC example I — Drell-Yan

Learned distribution matches truth



LHC example I — Drell-Yan

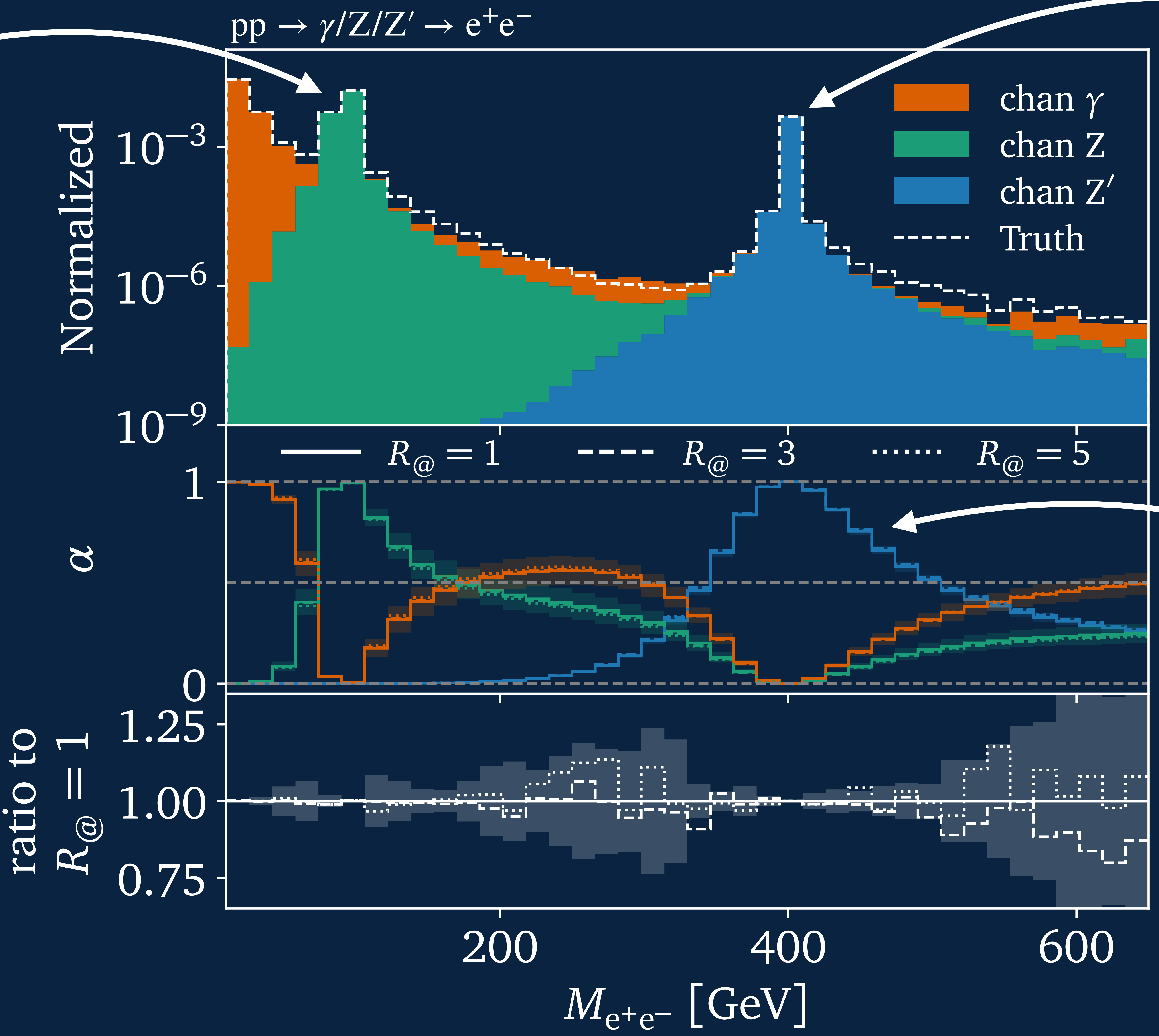
Learned distribution matches truth



Peaks mapped out by different channels

LHC example I — Drell-Yan

Learned distribution matches truth



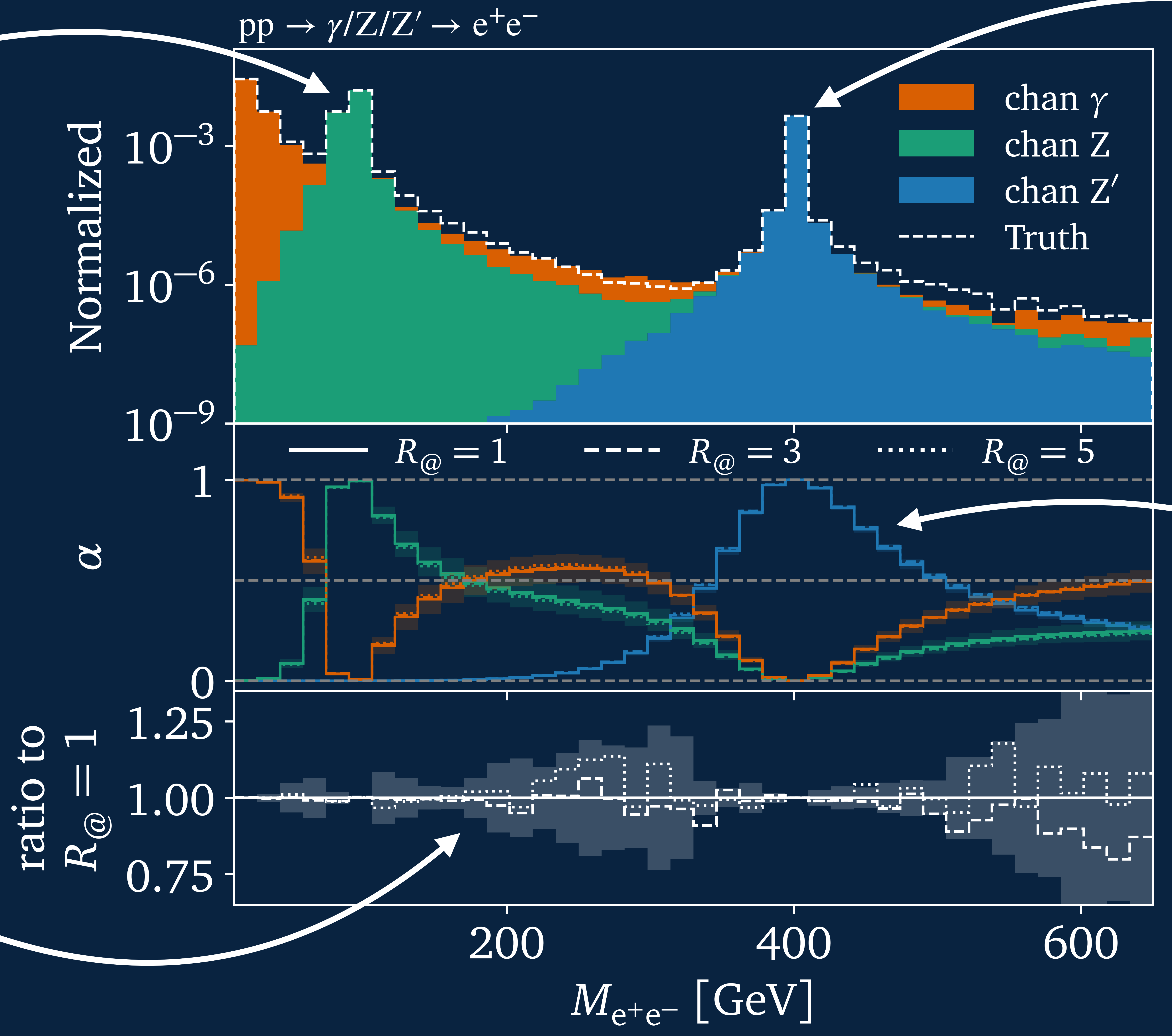
Peaks mapped out by different channels

Channel weights learned by network

LHC example I — Drell-Yan

Learned distribution matches truth

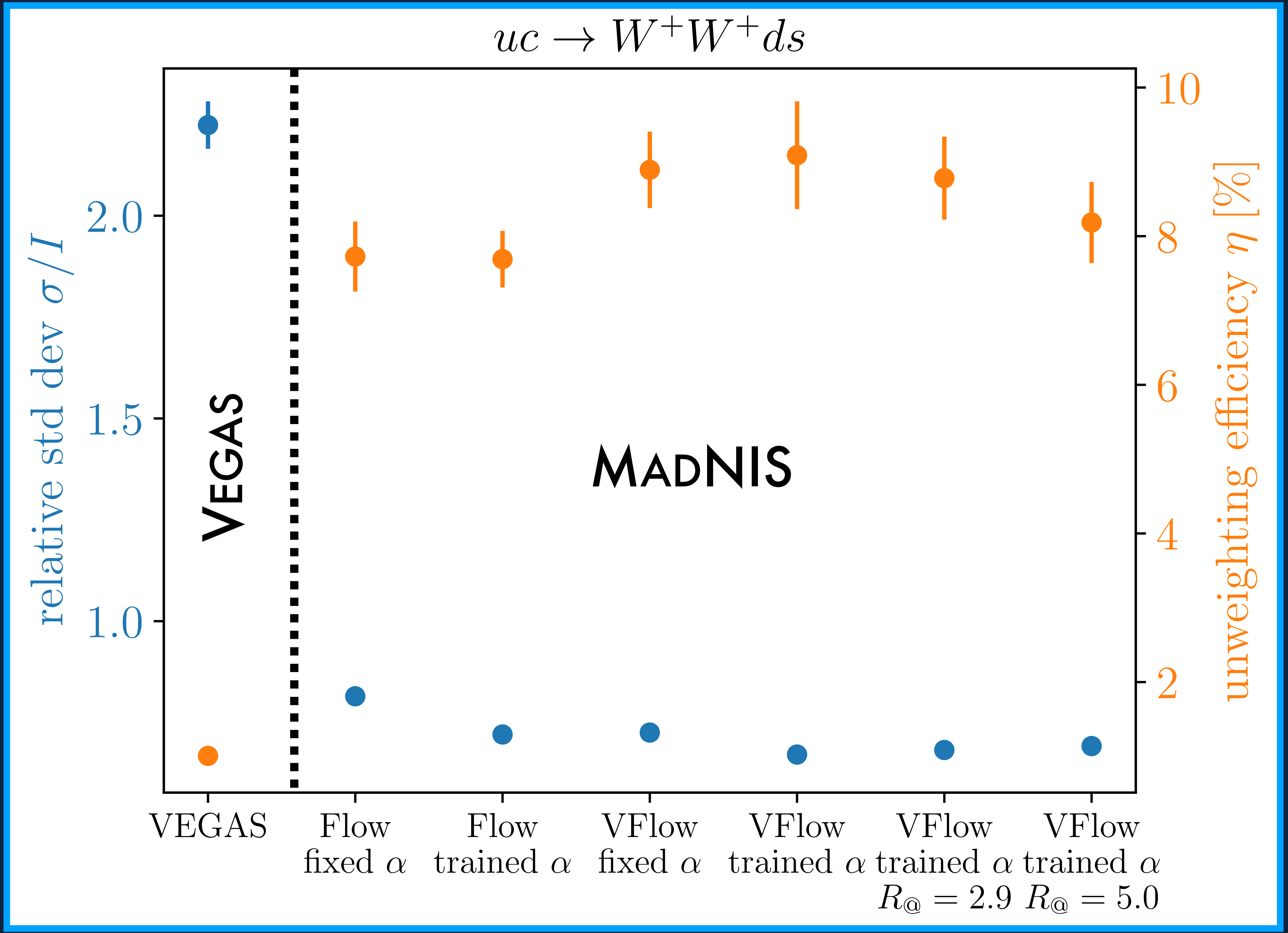
Peaks mapped out by different channels

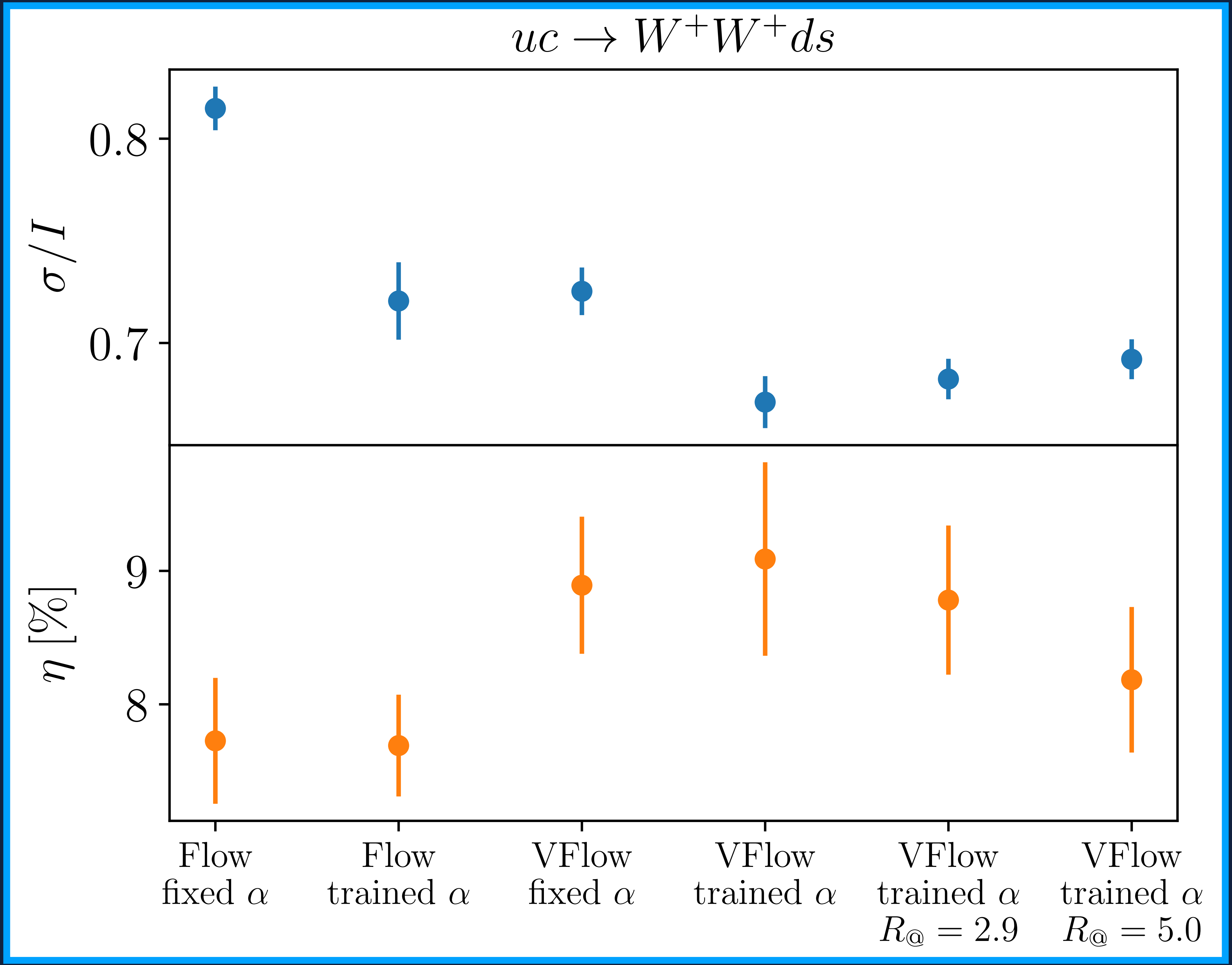


Channel weights learned by network

Use samples multiple times to make training faster

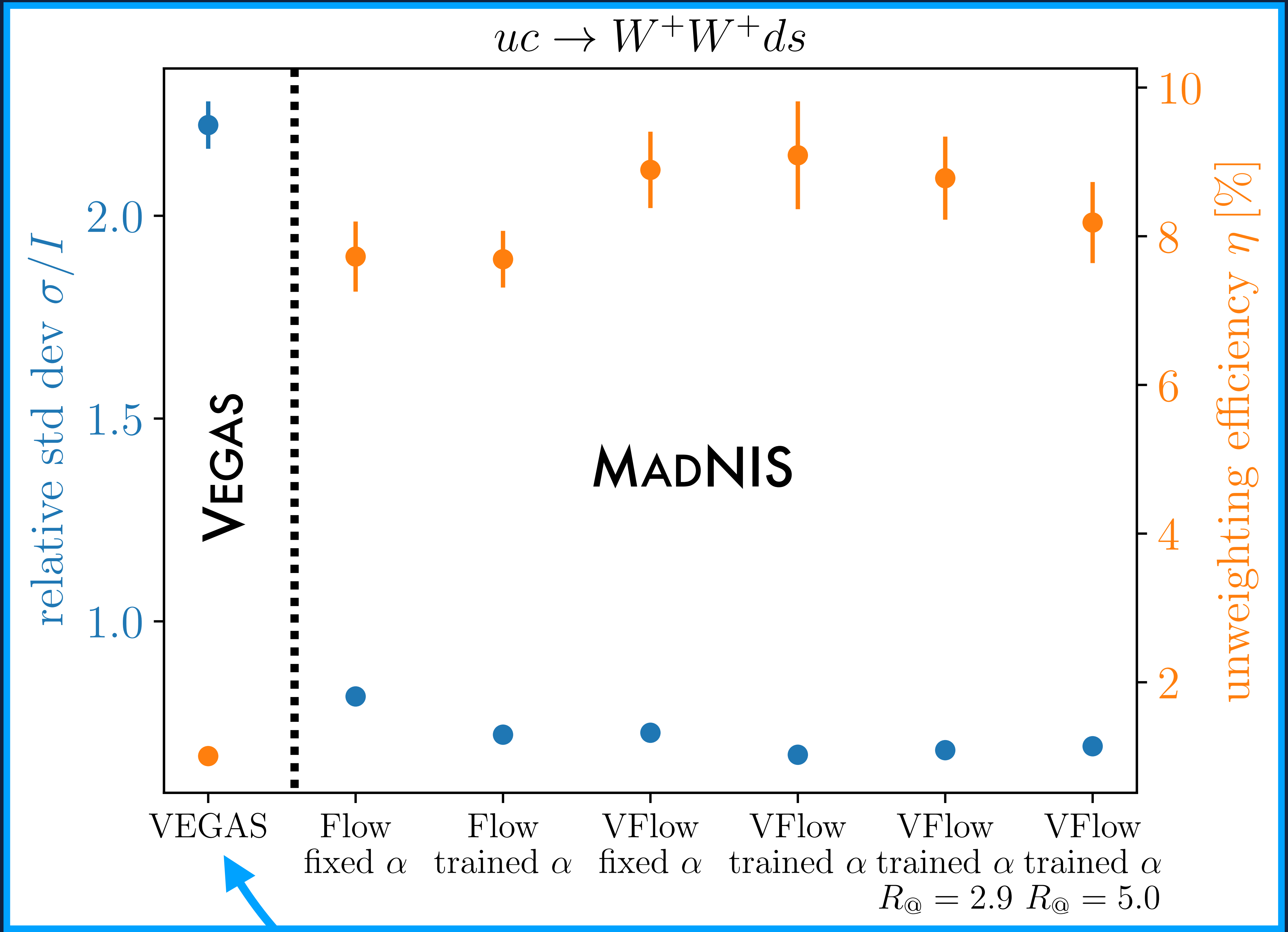
LHC example II — VBS

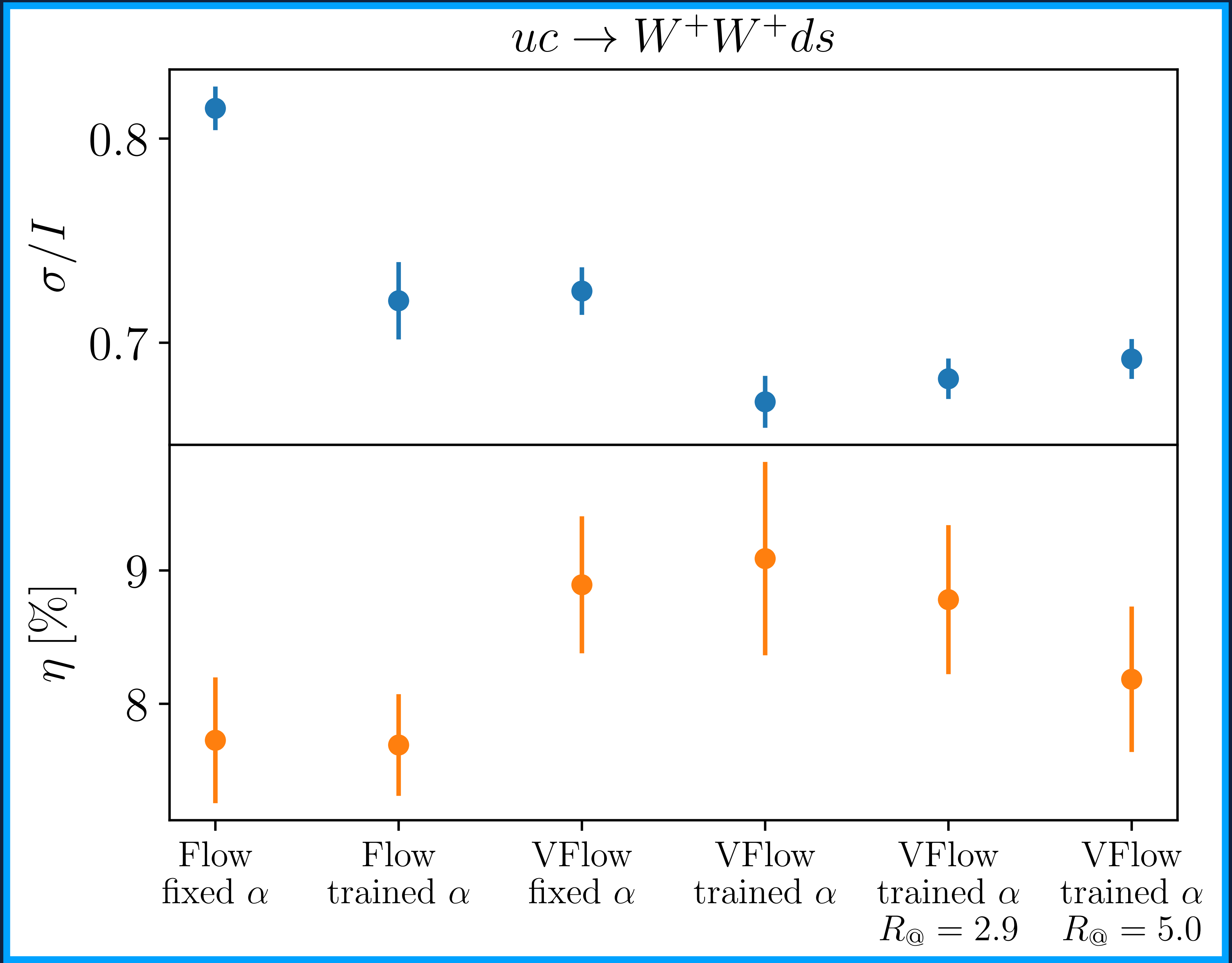




(preliminary)

LHC example II — VBS

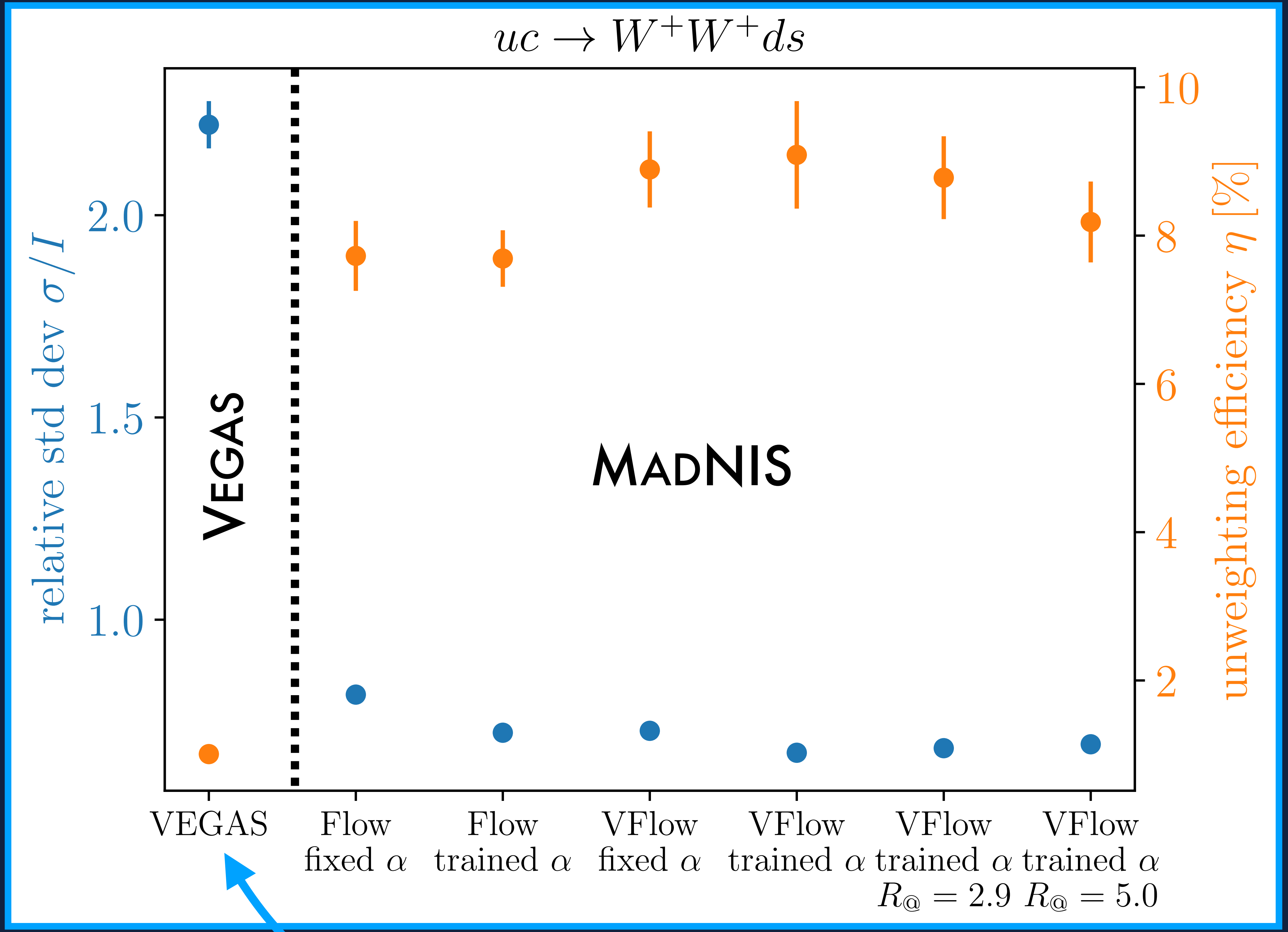


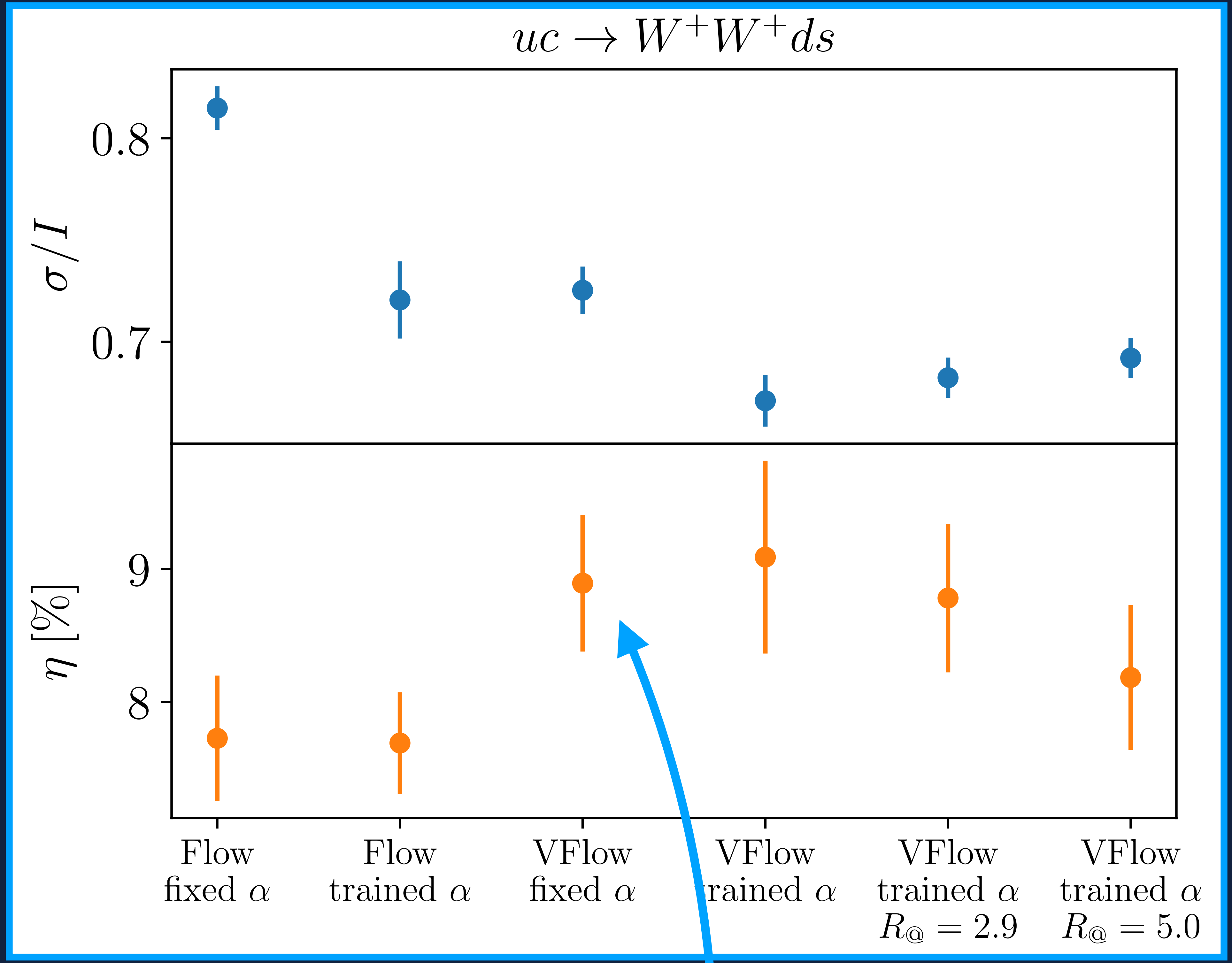


(preliminary)

Unweighting efficiency improved up to factor ~9 compared to VEGAS

LHC example II — VBS





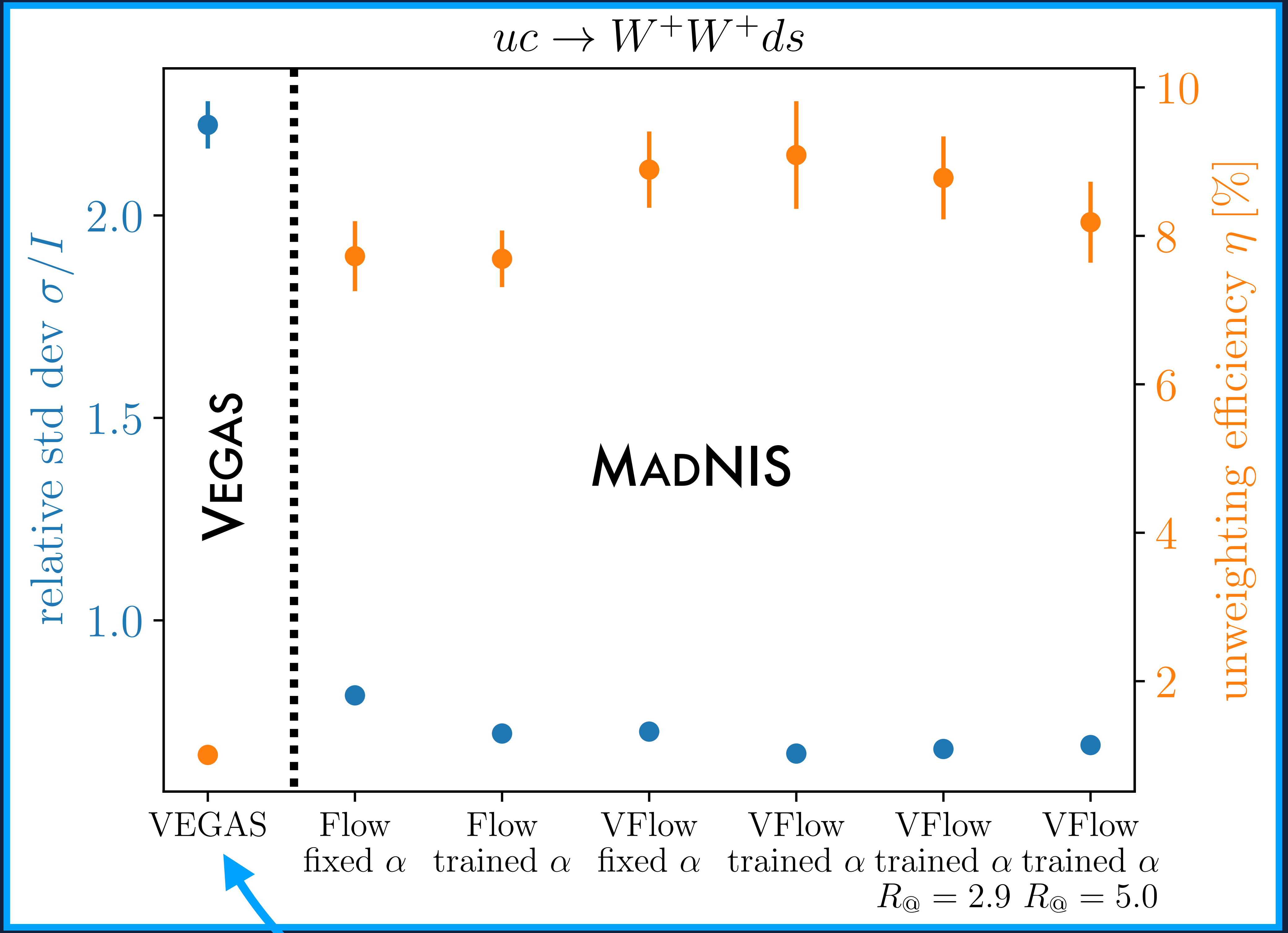
(preliminary)

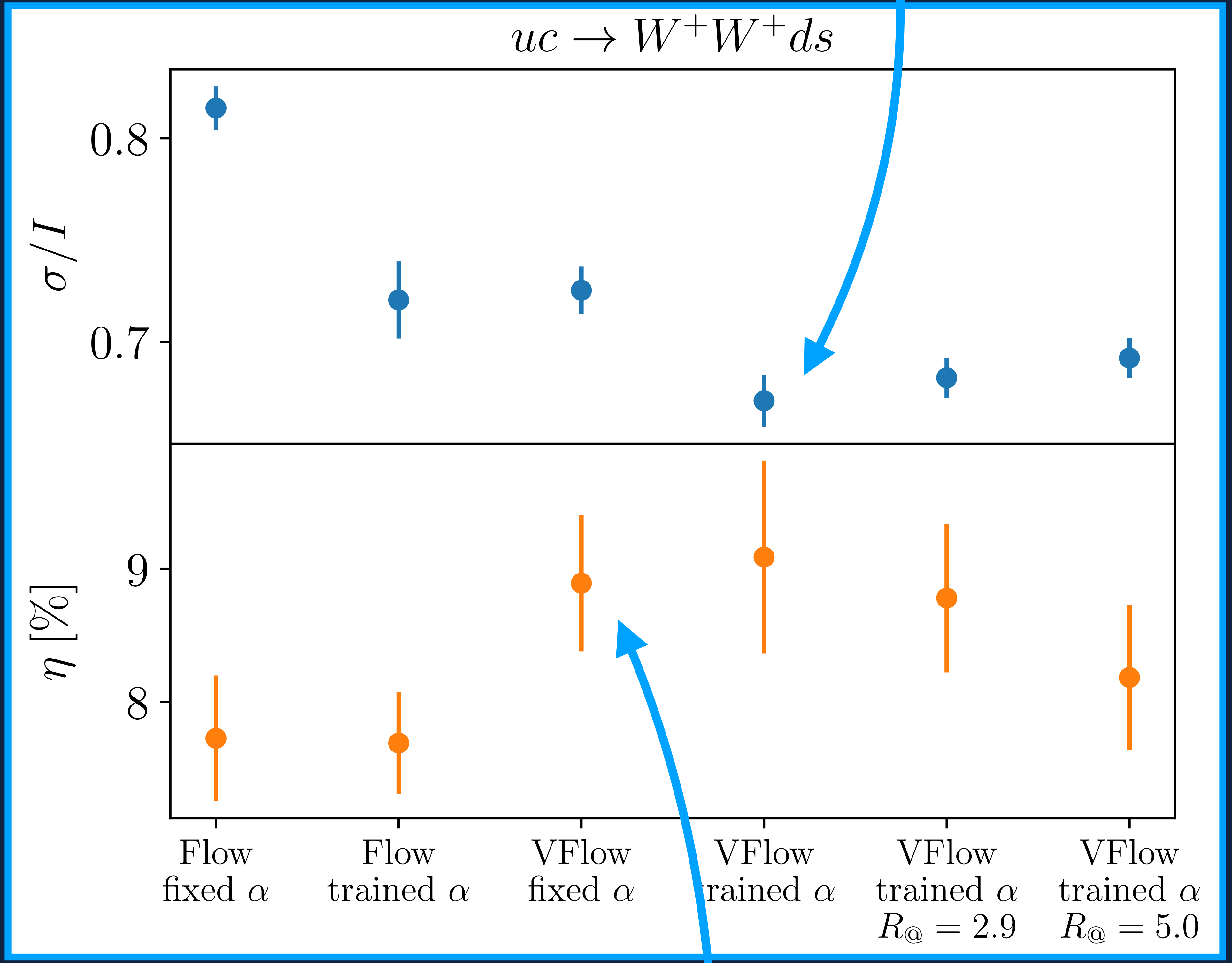
Unweighting efficiency improved up to factor ~9 compared to VEGAS

Big improvement from VEGAS initialization

LHC example II — VBS

Significant improvement from trained channel weights





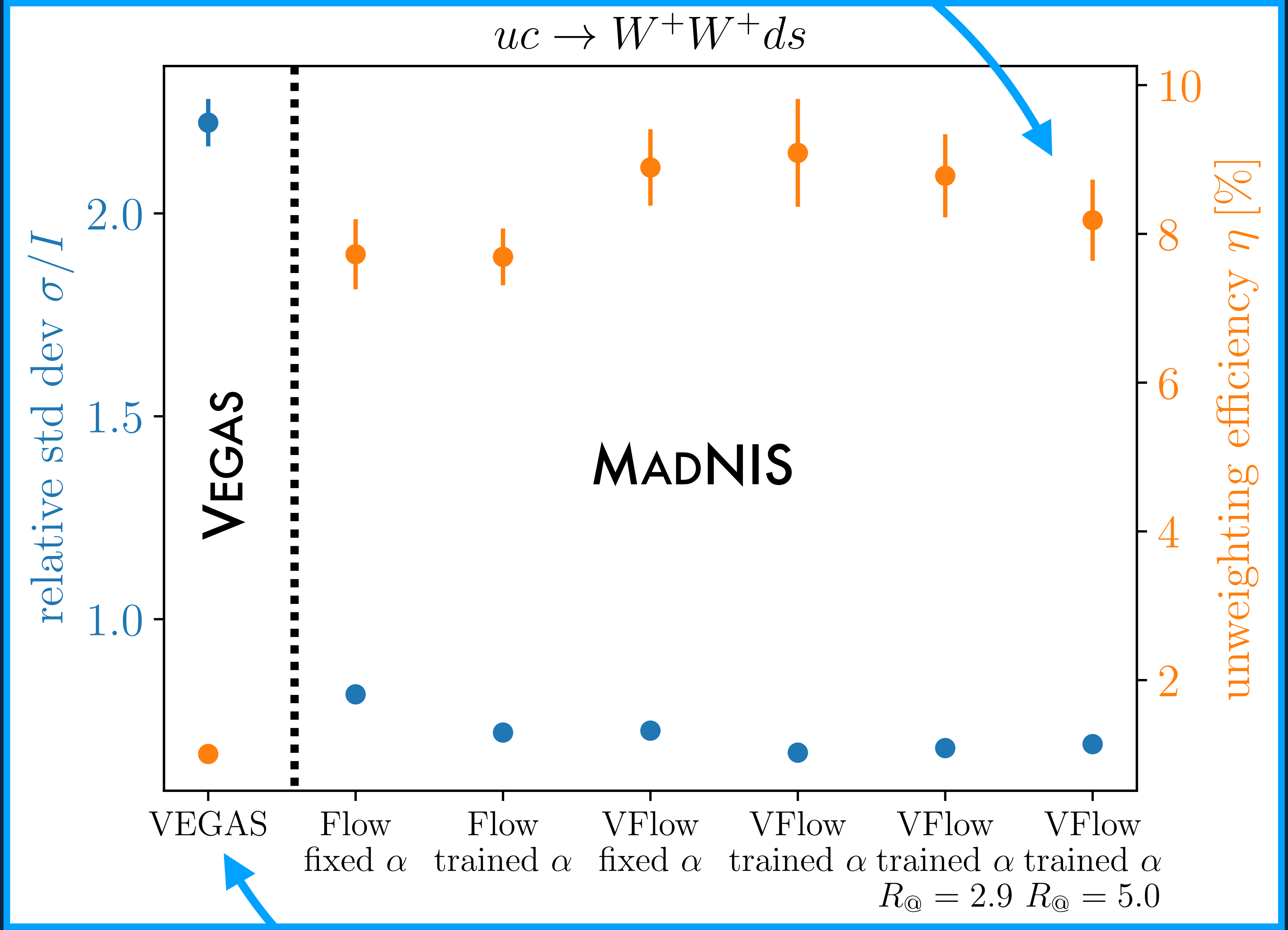
(preliminary)

Unweighting efficiency improved up to factor ~9 compared to VEGAS

Big improvement from VEGAS initialization

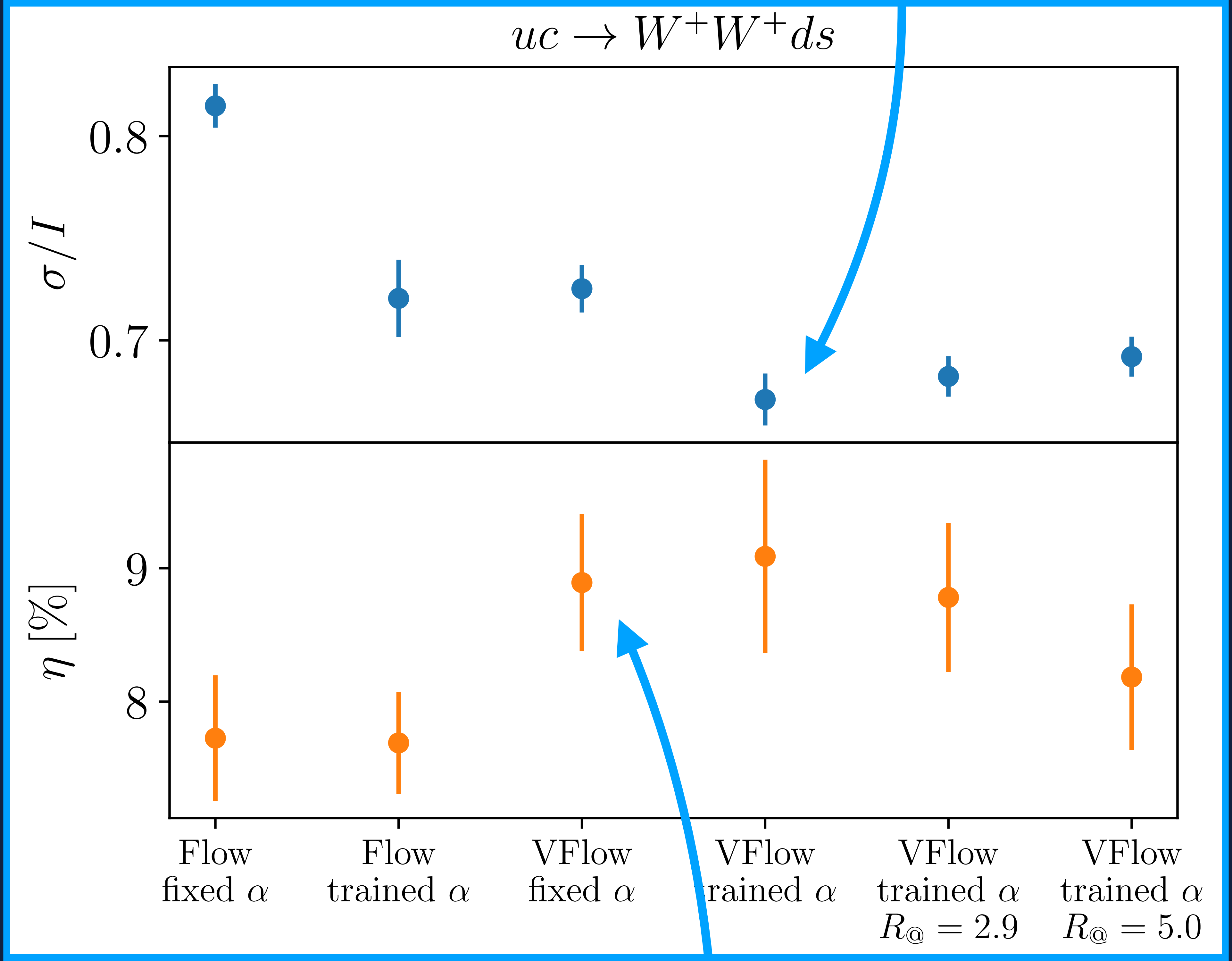
LHC example II — VBS

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

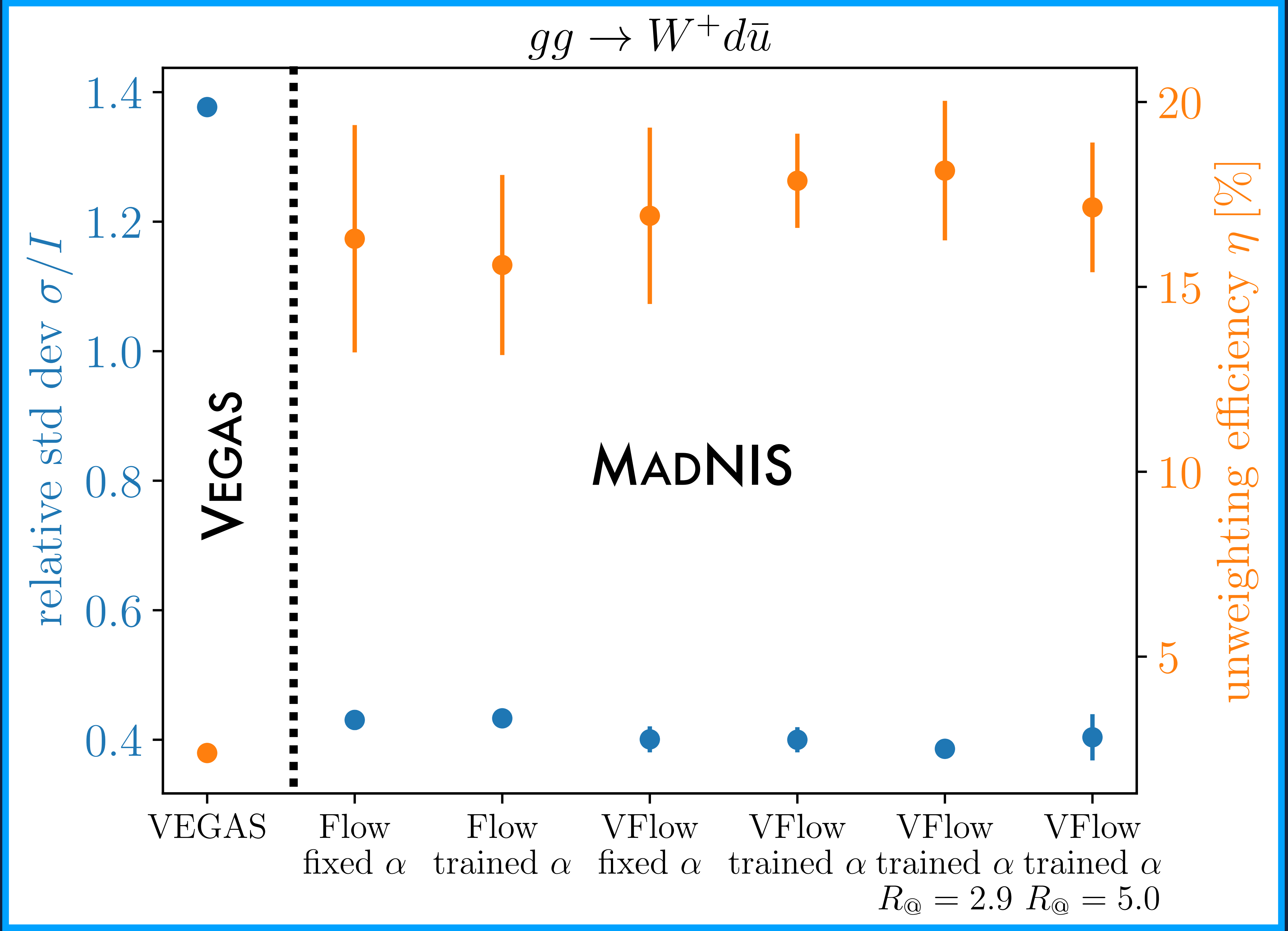


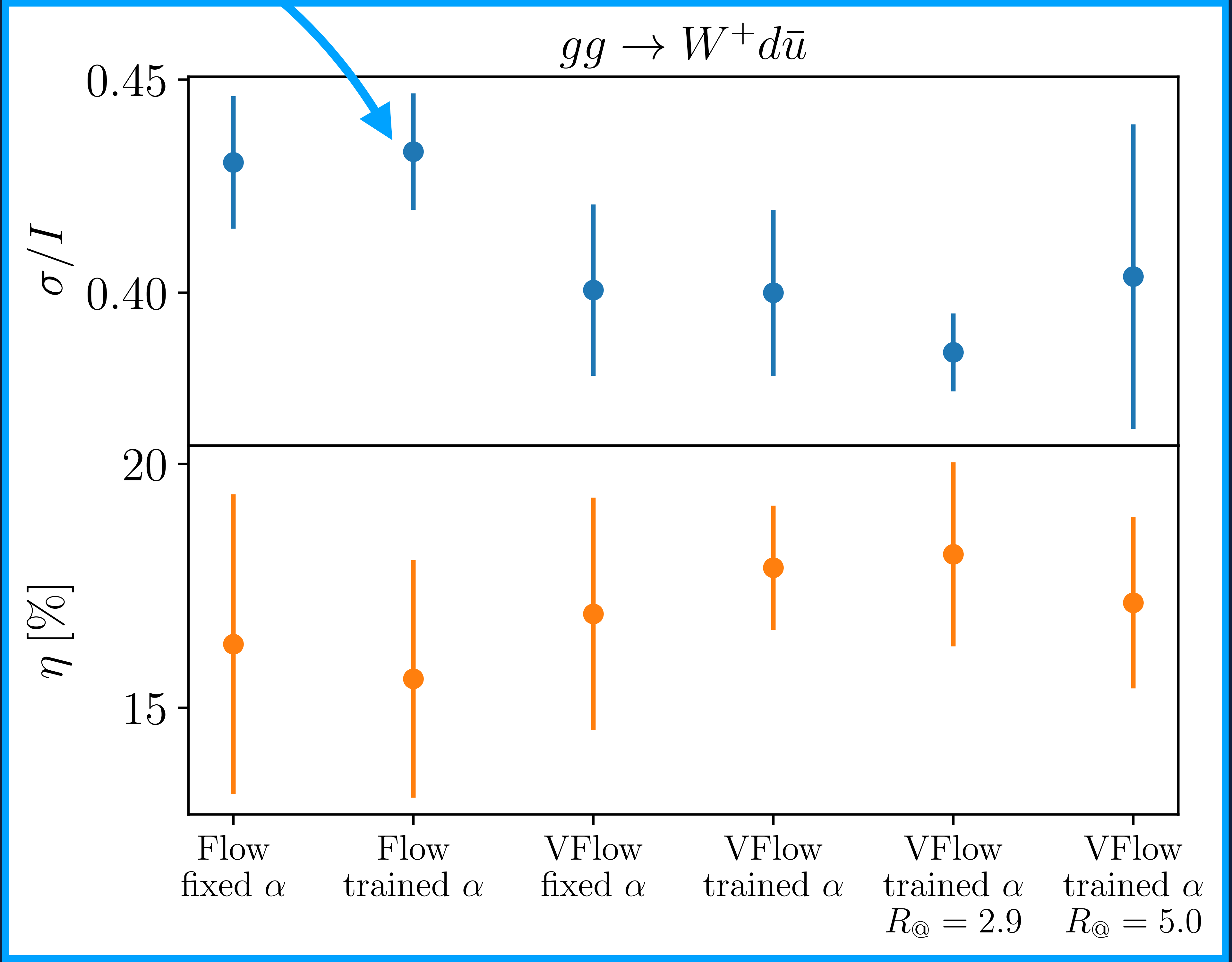
Big improvement from VEGAS initialization

(preliminary)

LHC example III — W + 2 jets

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





(preliminary)

Otherwise similar to results for VBS

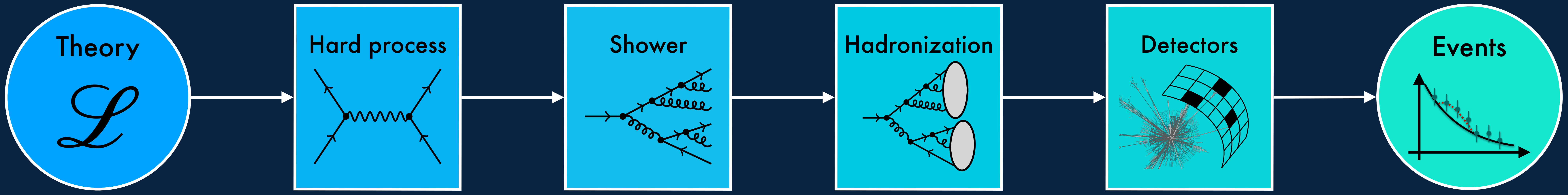
Summary and Outlook

Take-home message

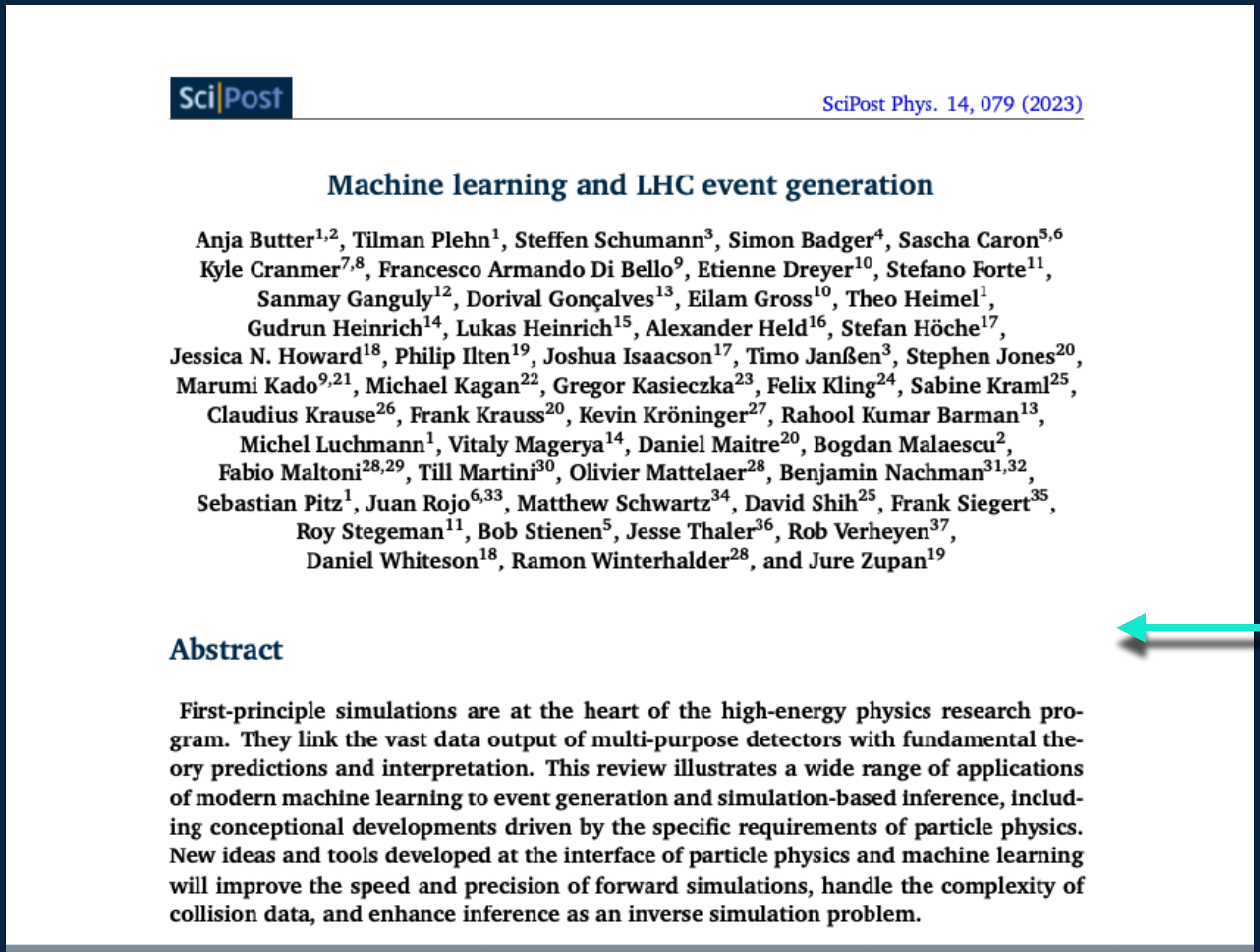
- **Fast** and **precise** predictions with ML-based simulations
- Normalizing flows provide statistically **well-defined likelihoods** for inference
- Account for **uncertainties** with **Bayesian neural networks**

Future exercises

- **Full integration** of ML-based simulations into standard tools → **MadGraph,….**
- Make everything run on the **GPU and differentiable** (MadJax - Heinrich et al. [2203.00057])



Summary and Outlook


SciPost | SciPost Phys. 14, 079 (2023)

Machine learning and LHC event generation

Anja Butter[1,2], Tilman Plehn[1], Steffen Schumann[3], Simon Badger[4], Sascha Caron[5,6], Kyle Cranmer[7,8], Francesco Armando Di Bello[9], Etienne Dreyer[10], Stefano Forte[11], Sanmay Ganguly[12], Dorival Gonçalves[13], Eilam Gross[10], Theo Heimel[1], Gudrun Heinrich[14], Lukas Heinrich[15], Alexander Held[16], Stefan Höche[17], Jessica N. Howard[18], Philip Ilten[19], Joshua Isaacson[17], Timo Janßen[3], Stephen Jones[20], Marumi Kado[9,21], Michael Kagan[22], Gregor Kasieczka[23], Felix Kling[24], Sabine Kraml[25], Claudius Krause[26], Frank Krauss[20], Kevin Kröninger[27], Rahool Kumar Barman[13], Michel Luchmann[1], Vitaly Magerya[14], Daniel Maitre[20], Bogdan Malaescu[2], Fabio Maltoni[28,29], Till Martini[30], Olivier Mattelaer[28], Benjamin Nachman[31,32], Sebastian Pitz[1], Juan Rojo[6,33], Matthew Schwartz[34], David Shih[25], Frank Siegert[35], Roy Stegeman[11], Bob Stienen[5], Jesse Thaler[36], Rob Verheyen[37], Daniel Whiteson[18], Ramon Winterhalder[28], and Jure Zupan[19]

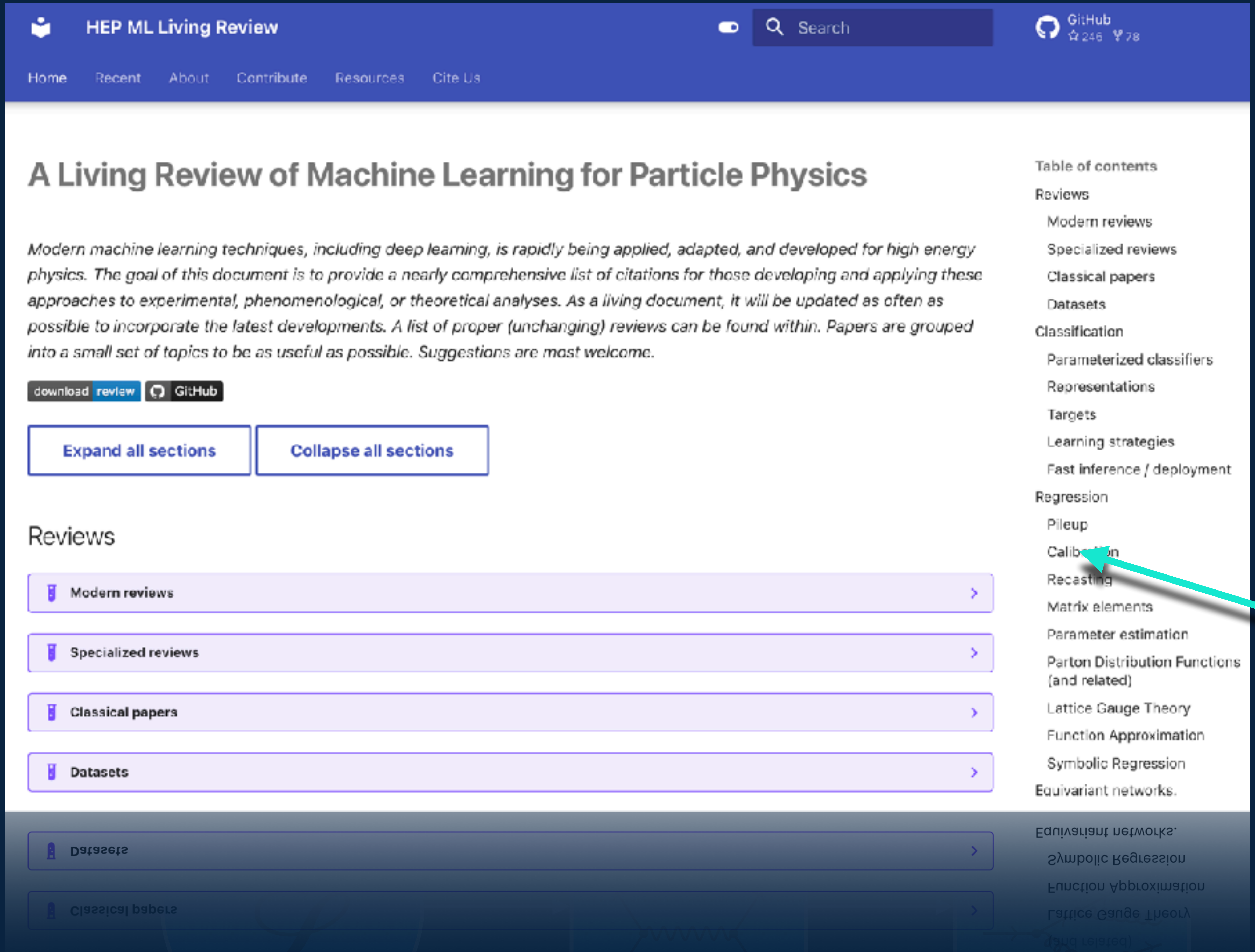
Abstract

First-principle simulations are at the heart of the high-energy physics research program. They link the vast data output of multi-purpose detectors with fundamental theory predictions and interpretation. This review illustrates a wide range of applications of modern machine learning to event generation and simulation-based inference, including conceptional developments driven by the specific requirements of particle physics. New ideas and tools developed at the interface of particle physics and machine learning will improve the speed and precision of forward simulations, handle the complexity of collision data, and enhance inference as an inverse simulation problem.

Future exercises

- **Full integration** of ML-based simulations into standard tools → **MadGraph,….**
- Make everything run on the **GPU and differentiable** (MadJax - Heinrich et al. [2203.00057])
- More details in our **Snowmass report**

Summary and Outlook



Future exercises

- **Full integration** of ML-based simulations into standard tools → **MadGraph,….**
- Make everything run on the **GPU and differentiable** (MadJax - Heinrich et al. [2203.00057])
- More details in our **Snowmass report**
- Stay tuned for many other **ML4HEP applications**