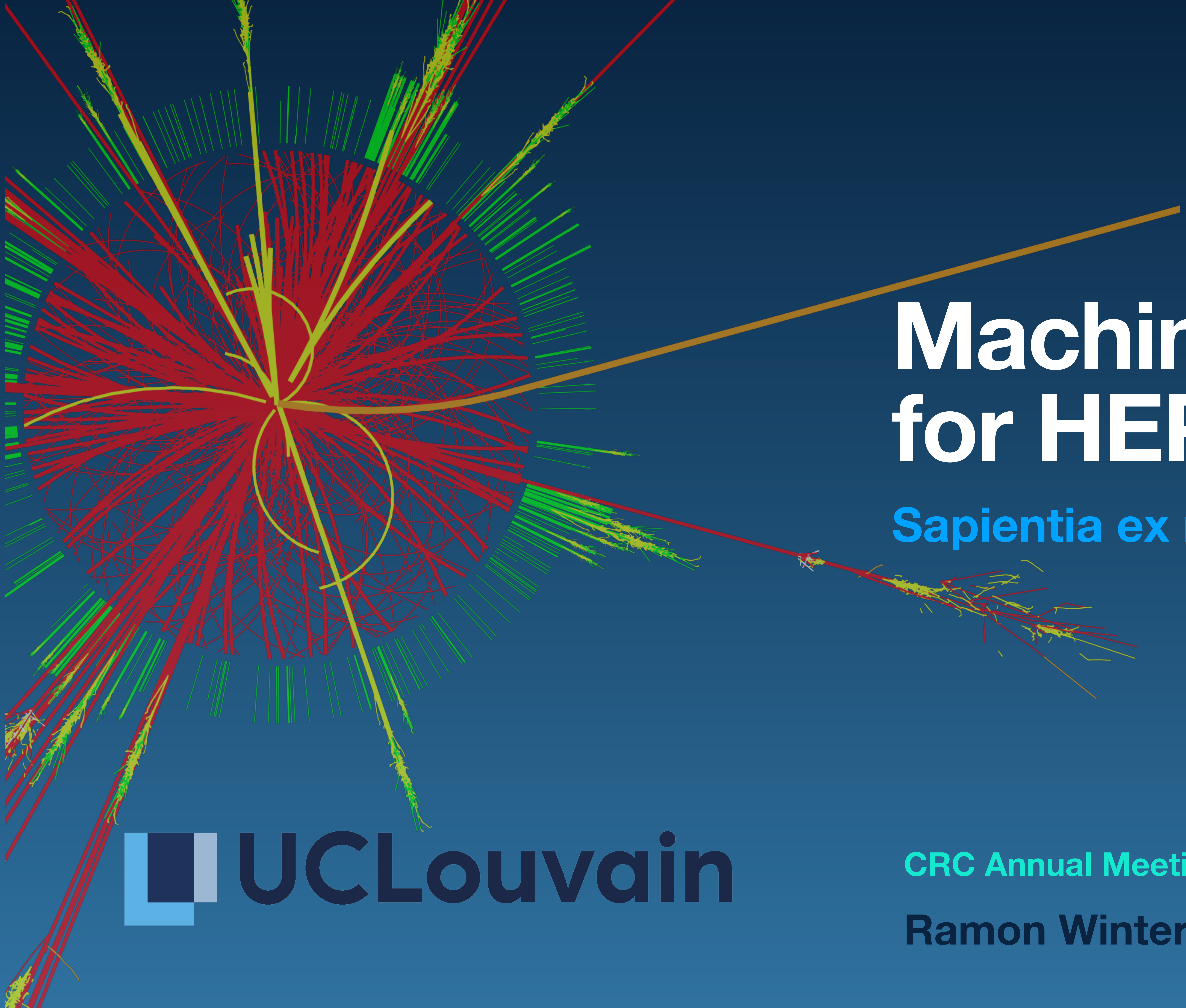


Machine Learning for HEP Theory

Sapientia ex machina?

CRC Annual Meeting — Aachen 2023

Ramon Winterhalder — UC Louvain

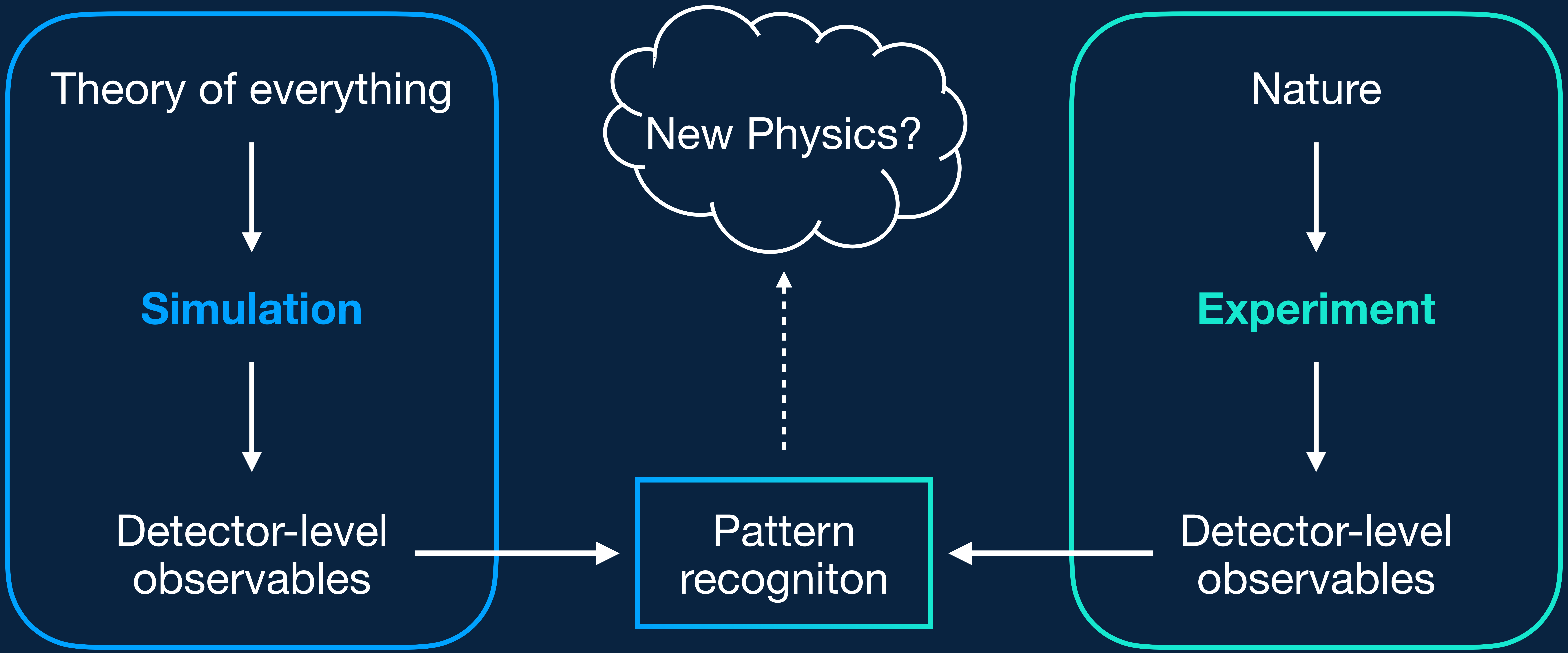
Why talk about machine learning?

Why talk about machine learning?

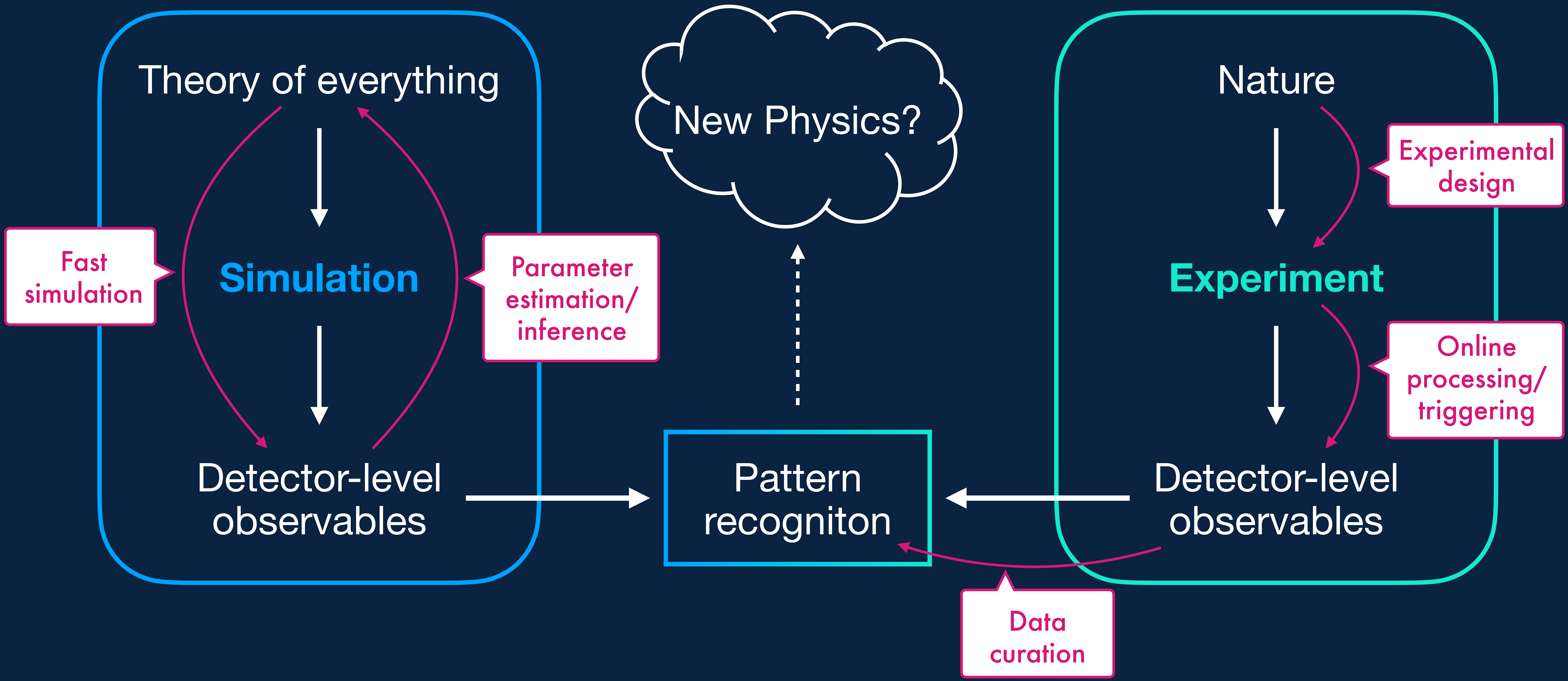
because

- rich toolbox of algorithms to develop expressive and flexible models for science
- fast development of new methods and algorithms in the past years
- promising applications in both theory and experiment
- large interest in HEP community: *IML, ML4Jets, MCnet, workshops,..*

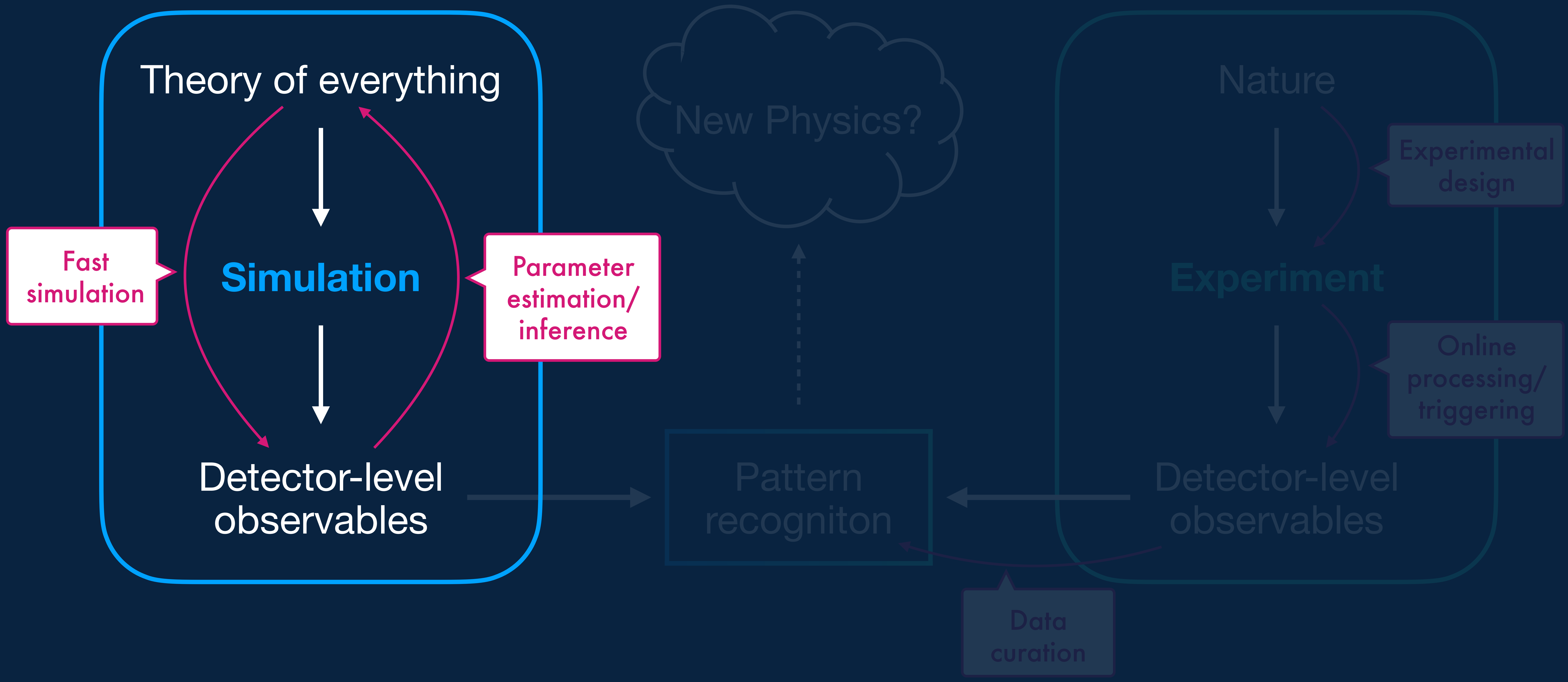
LHC analysis (oversimplified)



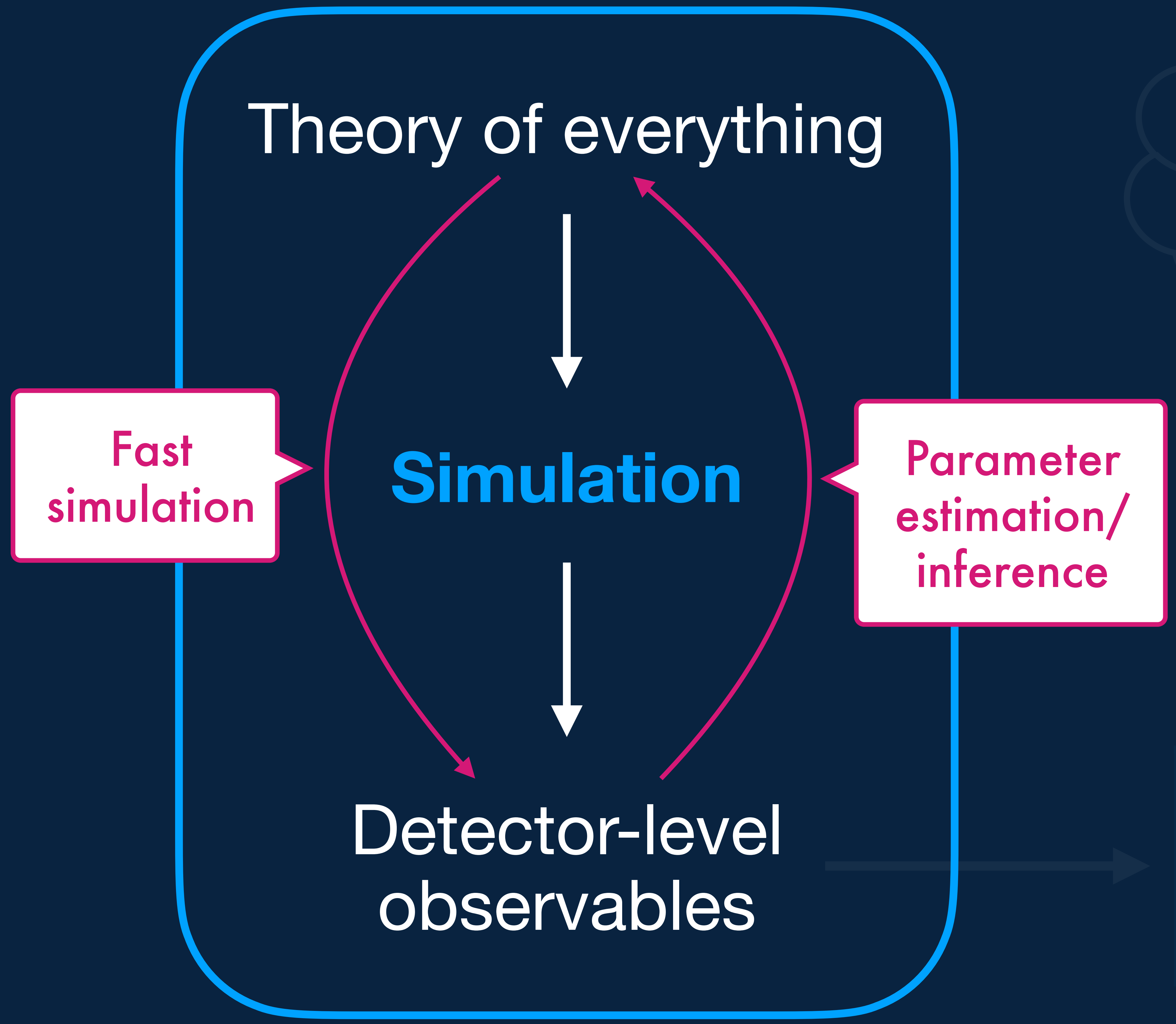
LHC analysis + ML



LHC analysis + ML

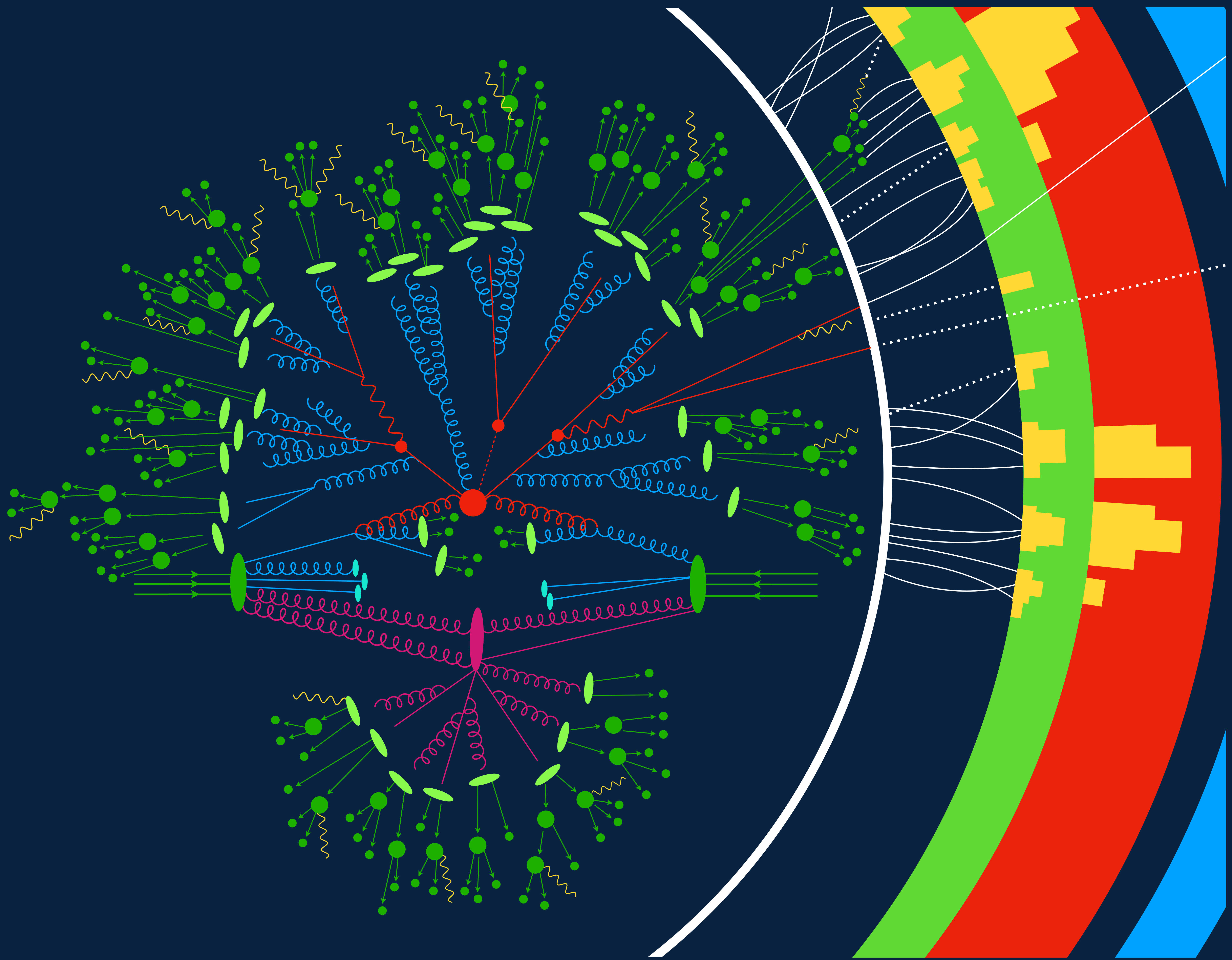


LHC analysis + ML

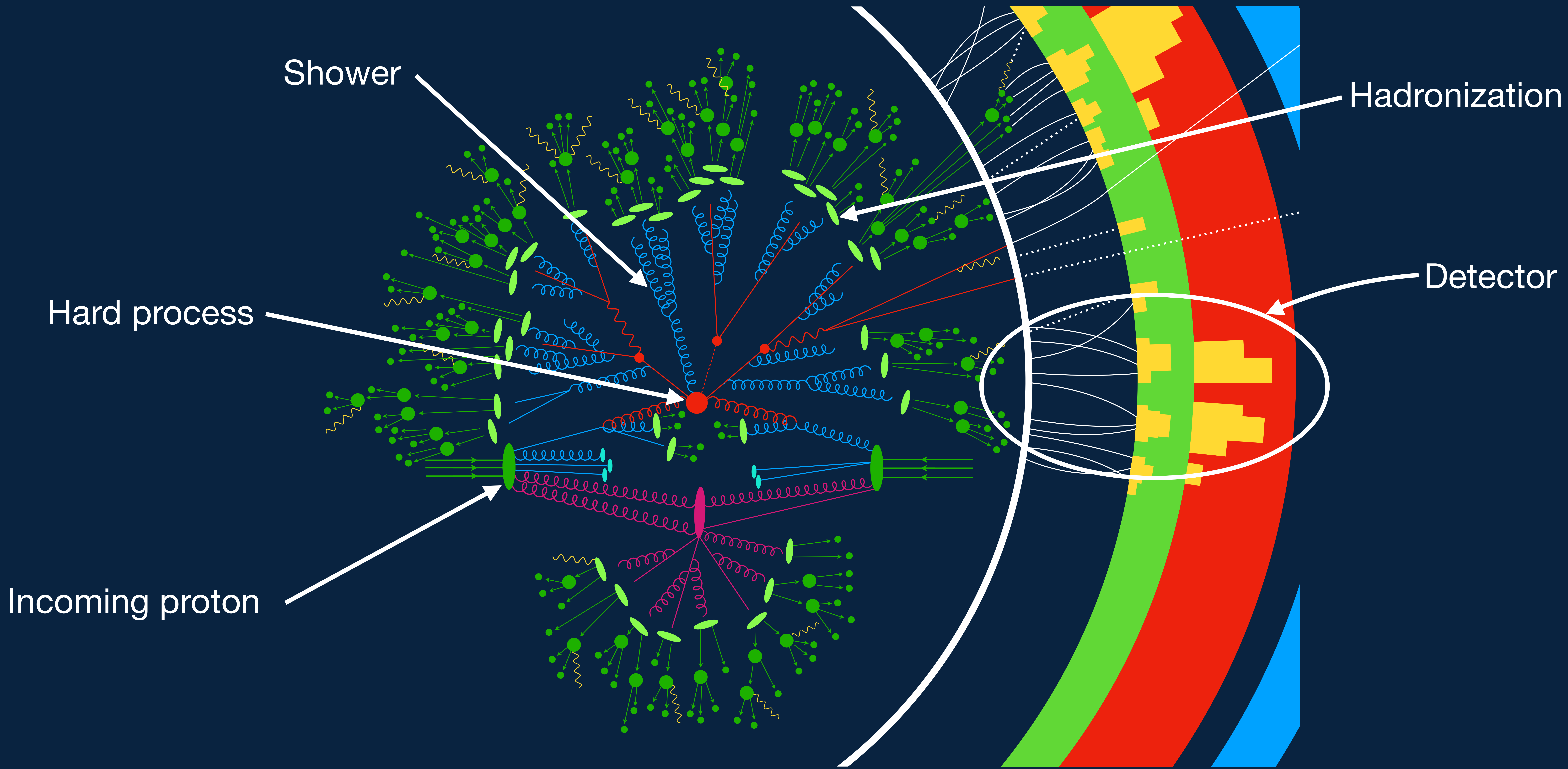


How to simulate LHC events?

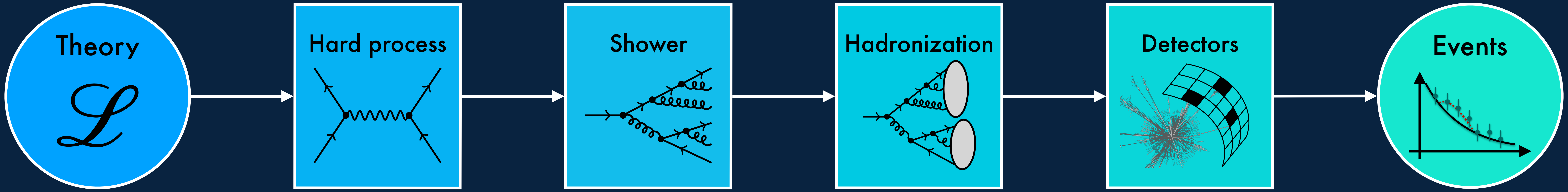
How to simulate LHC events



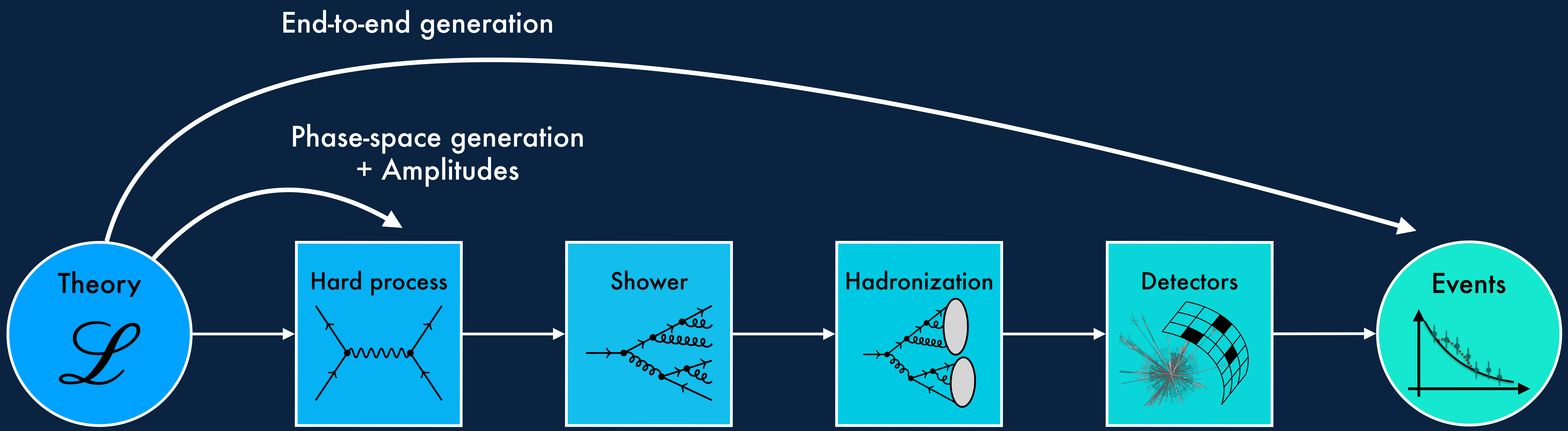
How to simulate LHC events



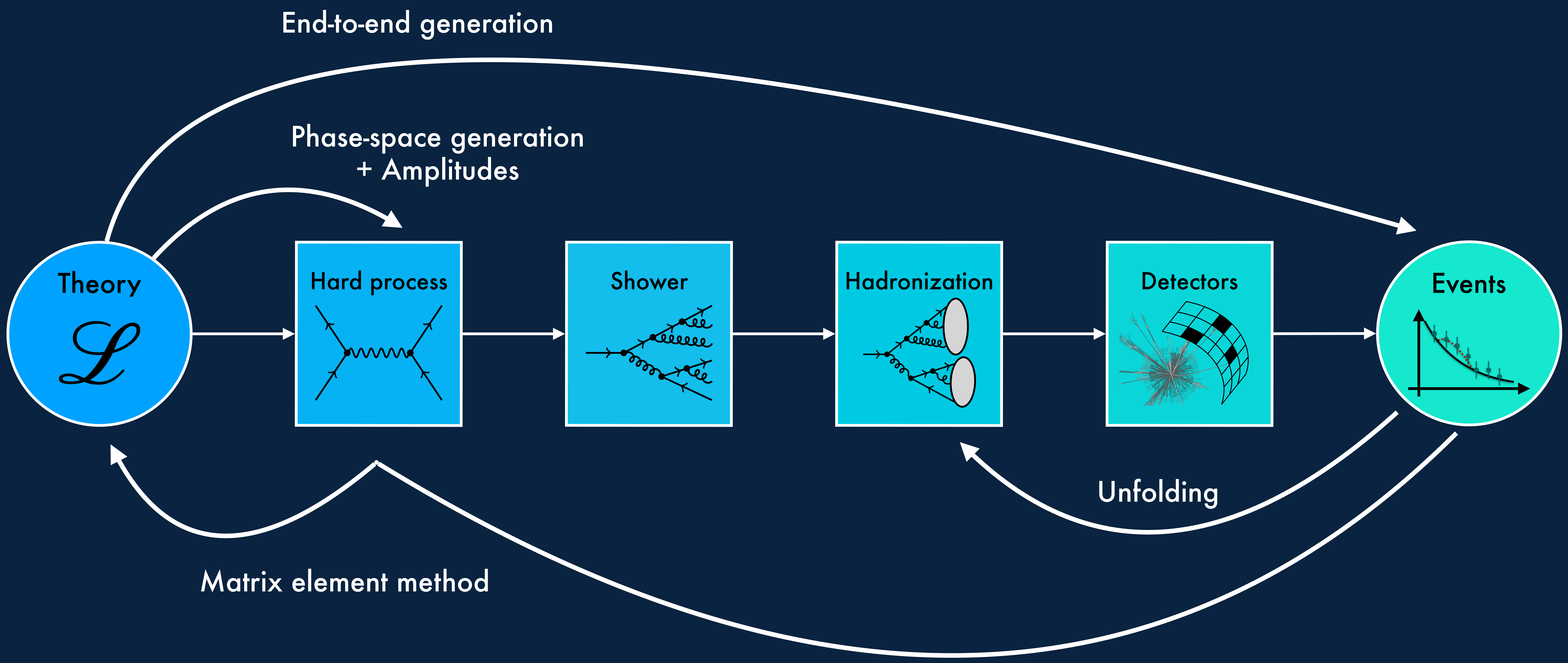
ML aided simulation chain



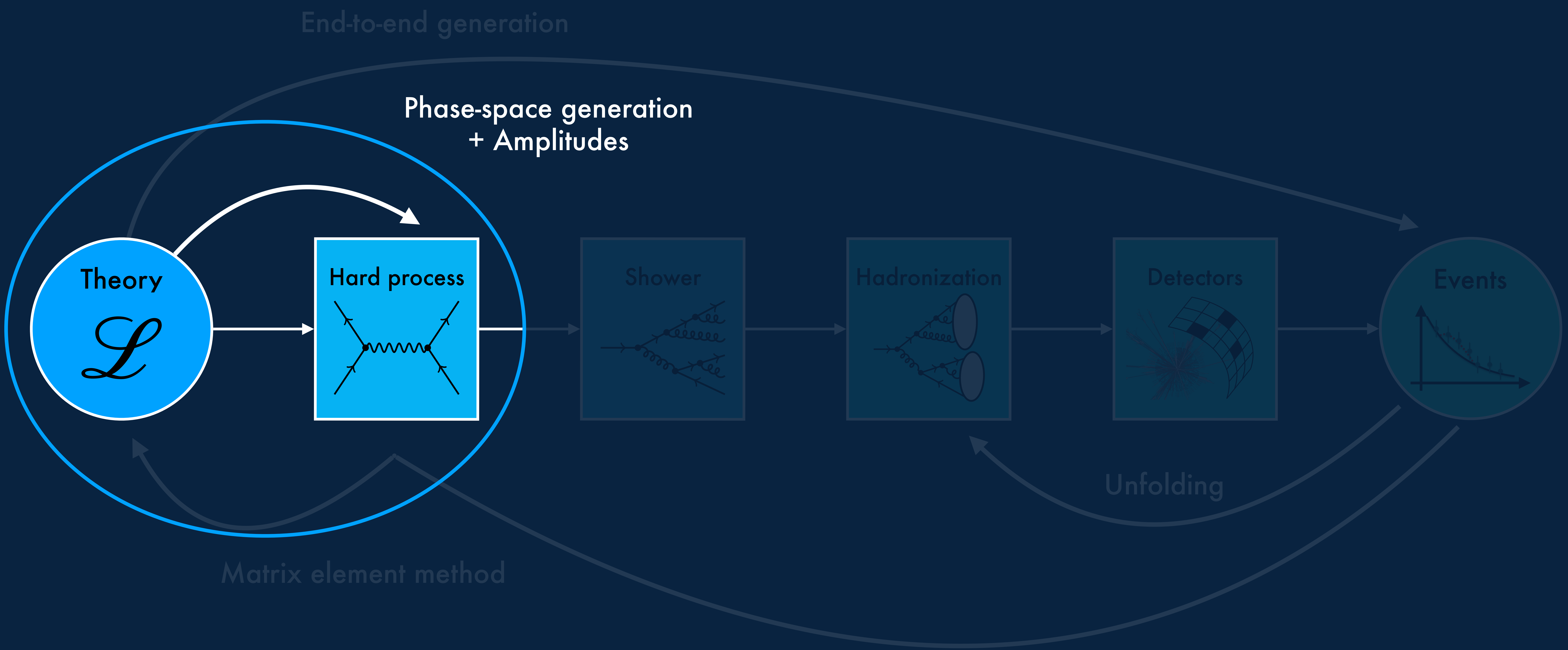
ML aided simulation chain



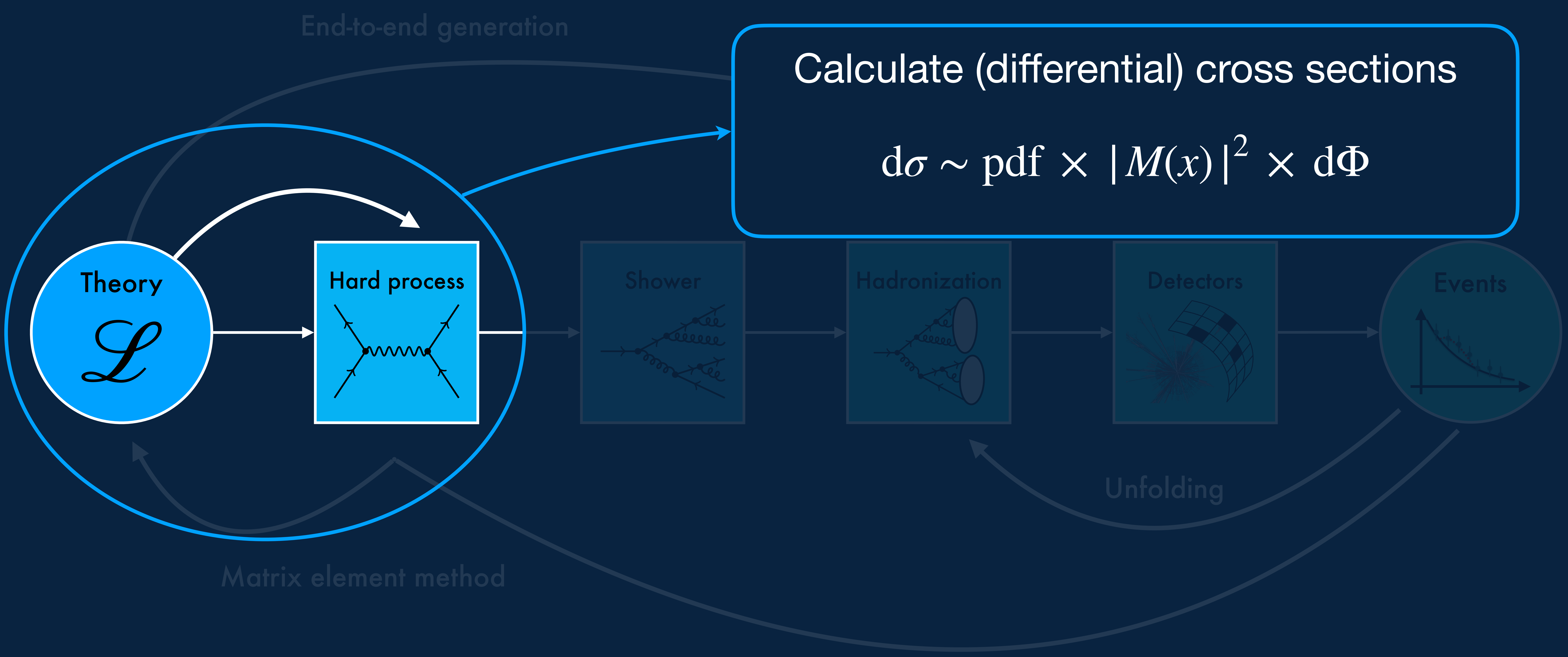
ML aided simulation chain



ML aided simulation chain



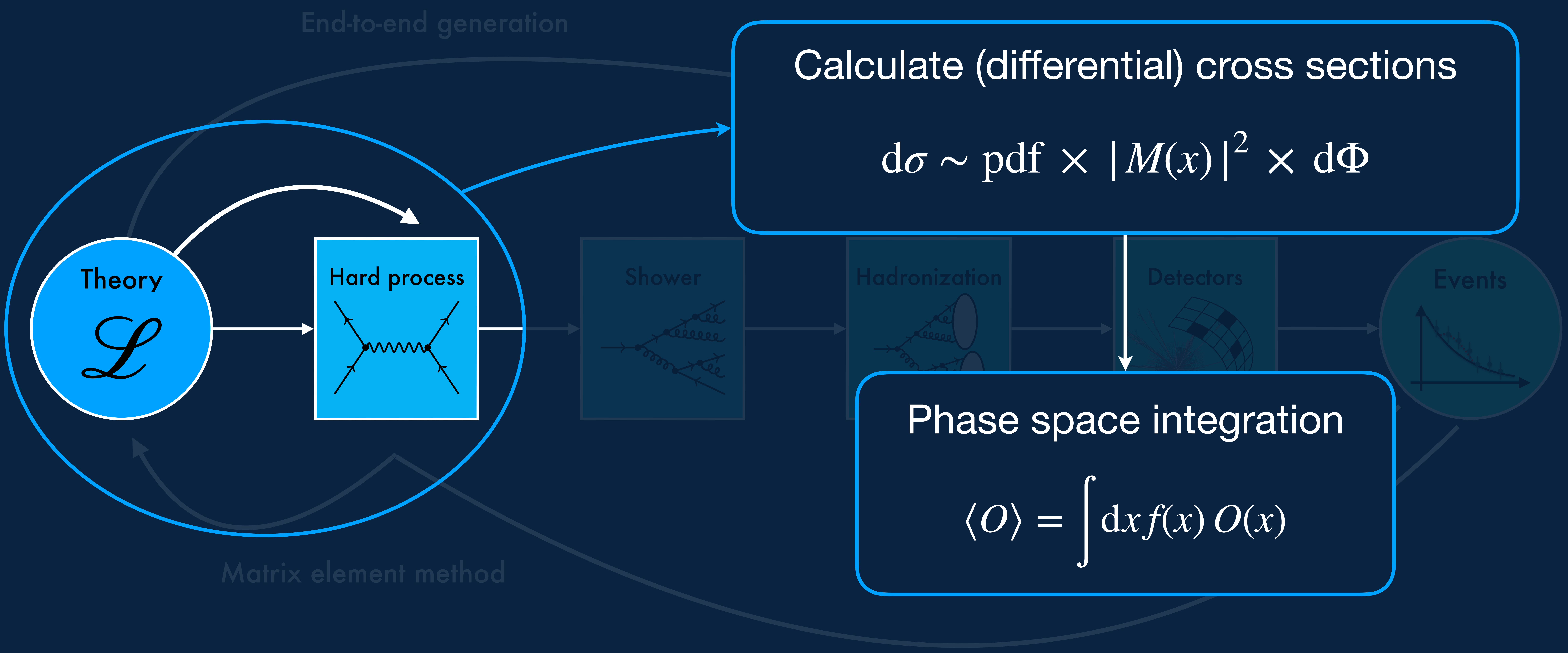
ML aided simulation chain



Calculate (differential) cross sections

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

ML aided simulation chain



Are there bottlenecks?

Are there bottlenecks?

Yes! Because

- Analytic integration not feasible: PDFs, cuts, jet algorithm, complex amplitudes, …
- Another problem is the high-dimensionality of the integrand
- Standard numerical methods scale badly: error $\sim N^{-2/D} \ldots N^{-4/D}$
- Use Monte Carlo integration instead: error $\sim N^{-1/2}$

Are there bottlenecks?

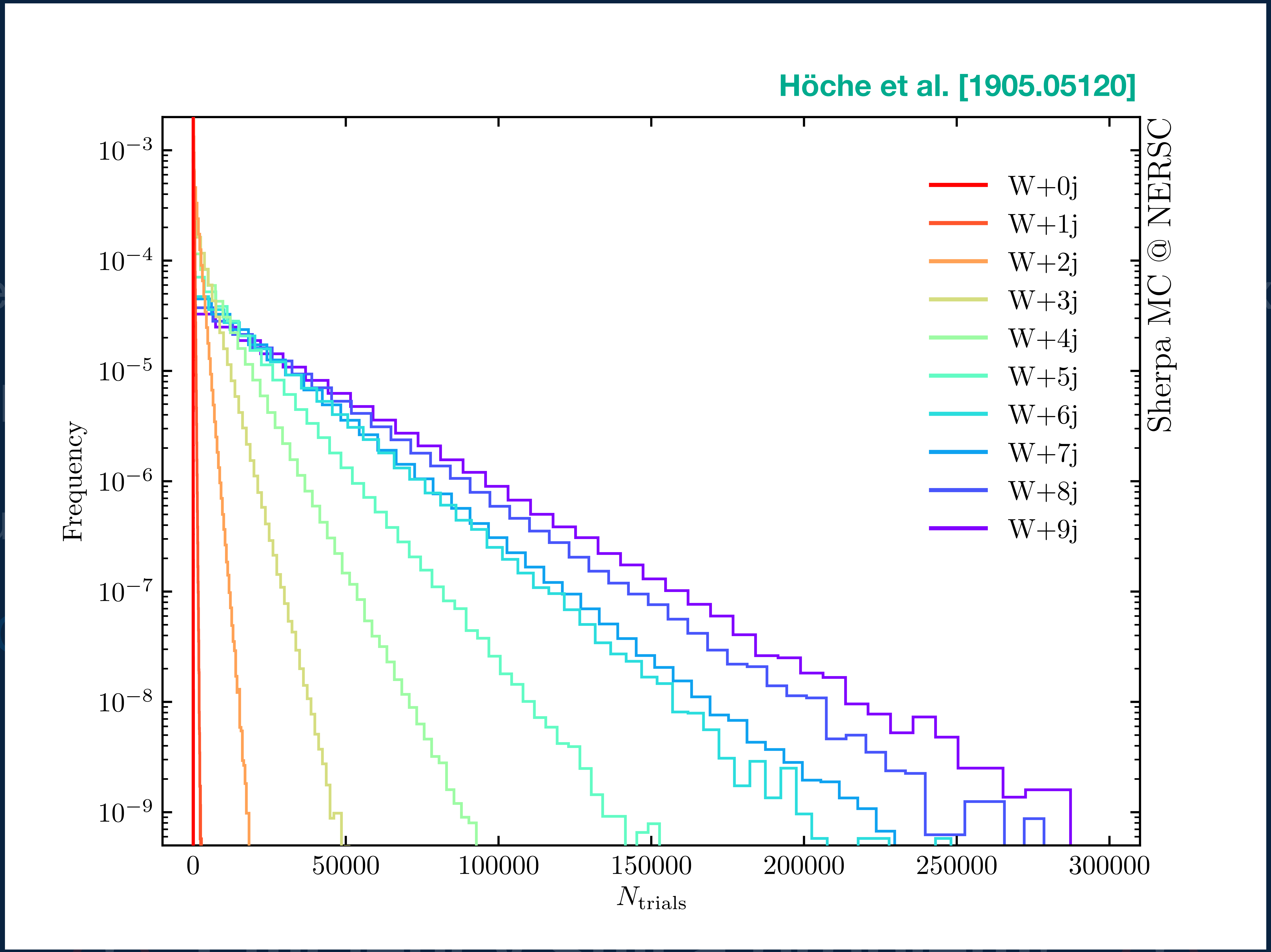
Yes! Because

- Analytic integration not feasible: PDFs, cuts, jet algorithm, complex amplitudes, …
- Another problem is the high-dimensionality of the integrand
- Standard numerical methods scale badly: error $\sim N^{-2/D} \dots N^{-4/D}$
- Use Monte Carlo integration instead: error $\sim N^{-1/2}$

⚠ Efficiency still a problem! ⚠

Are there bottlenecks?

- Analytic inte… amplitudes, …
- Another pro…
- Standard nu…
- Use Monte C…



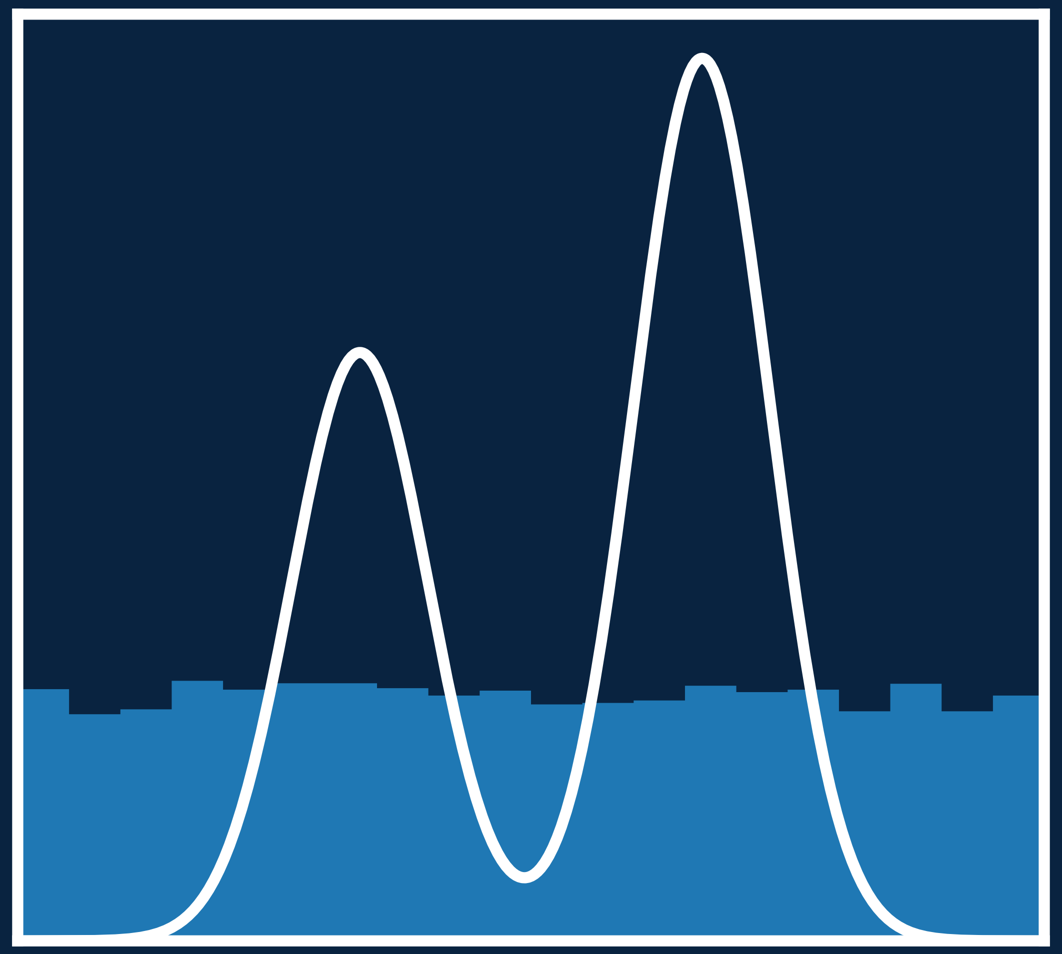
Efficiency still a problem!

Monte Carlo integration

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

Monte Carlo integration

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

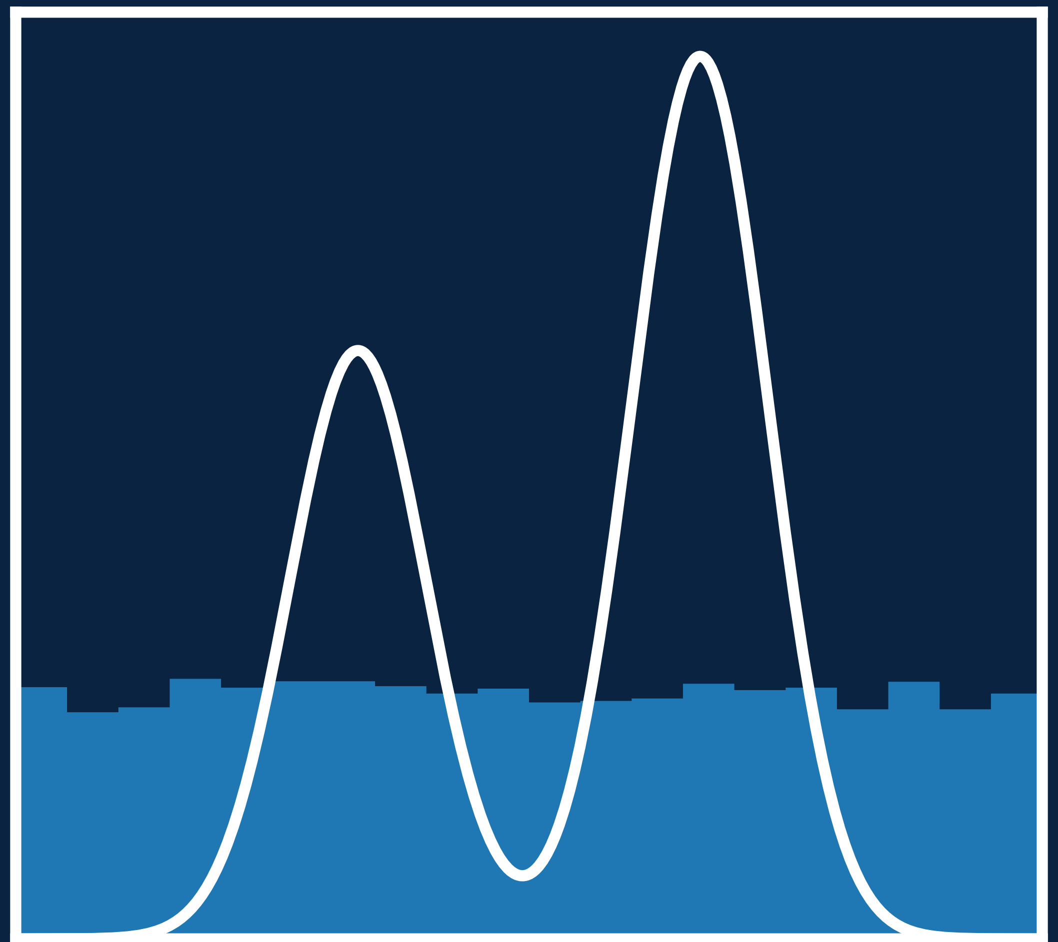


Flat sampling:
inefficient

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

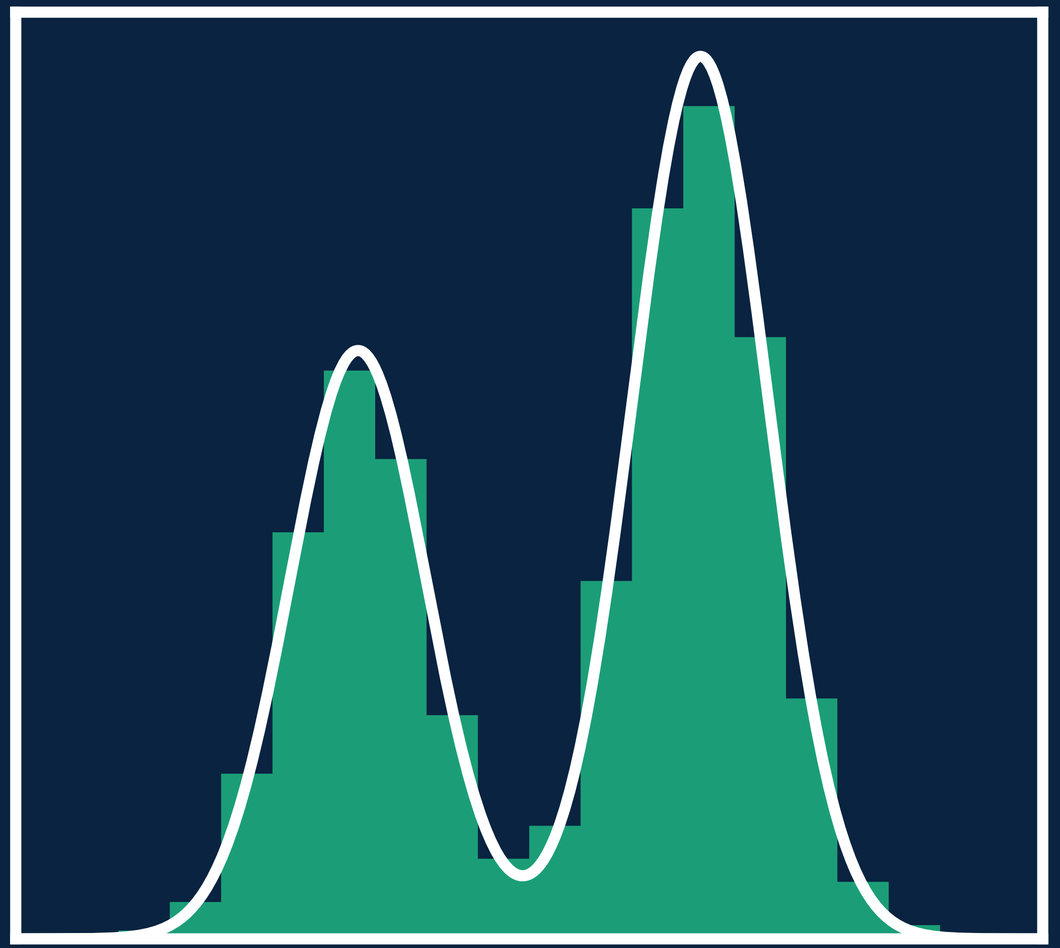
Monte Carlo integration

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



Flat sampling:
inefficient

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

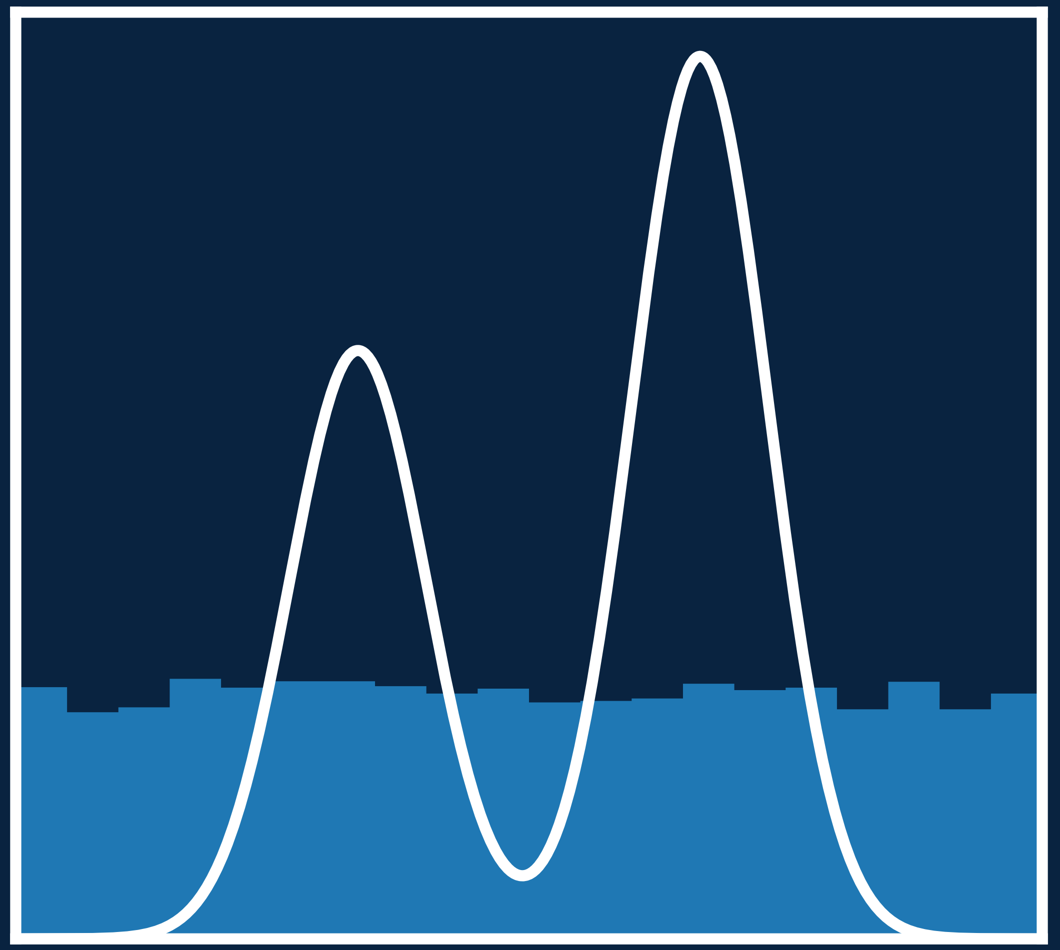


Importance sampling:
find g close to f

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

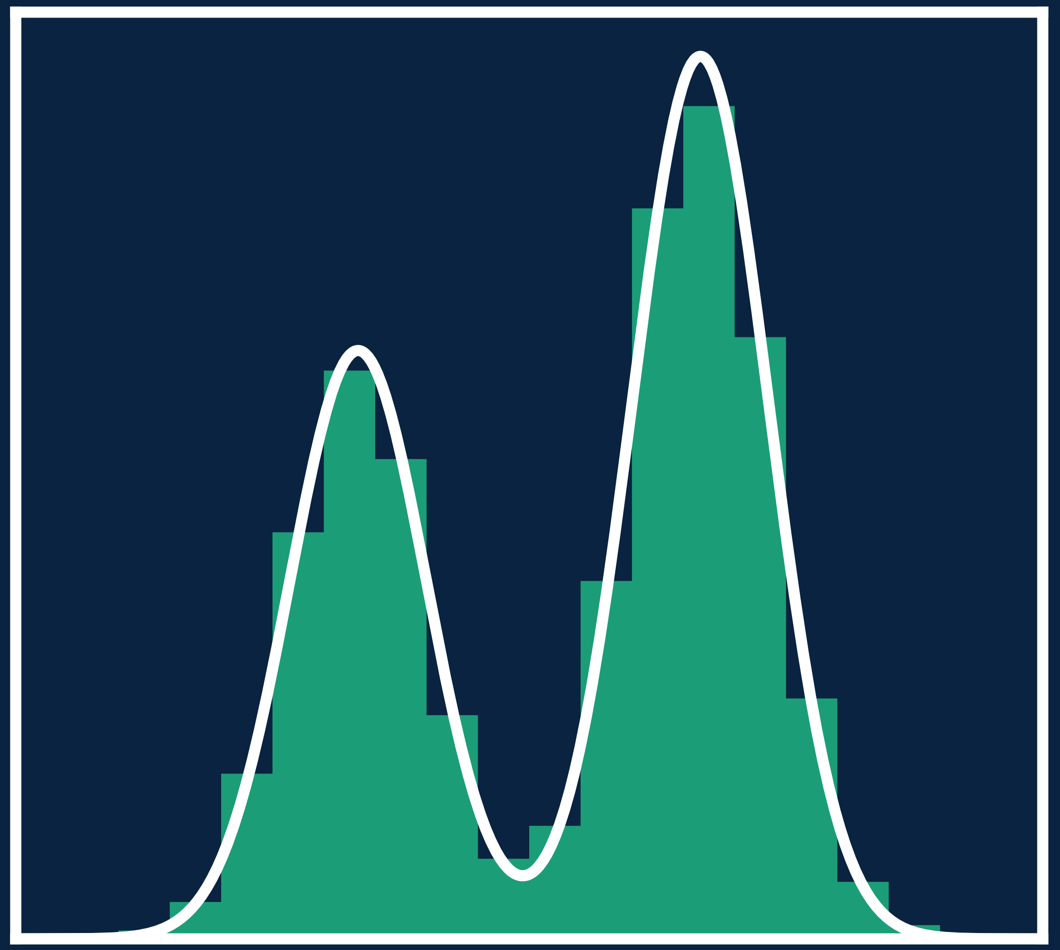
Monte Carlo integration

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



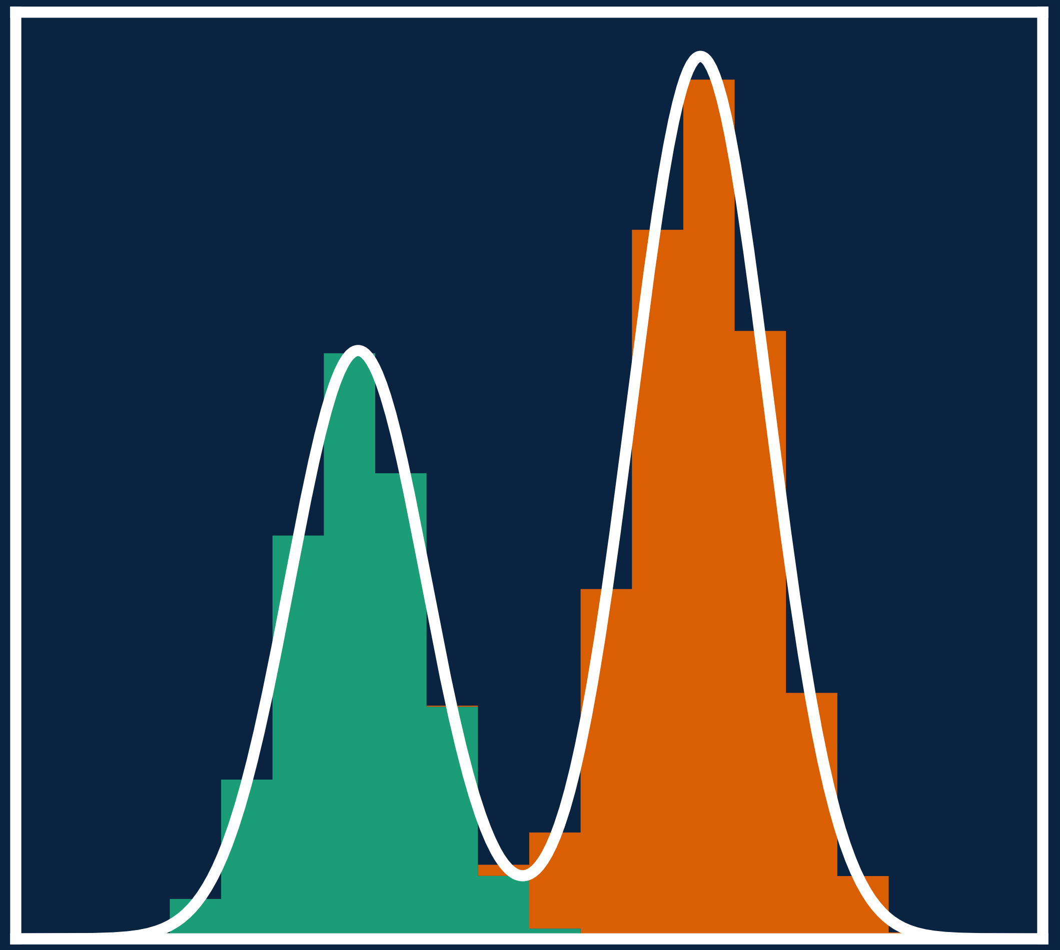
Flat sampling:
inefficient

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



Importance sampling:
find g close to f

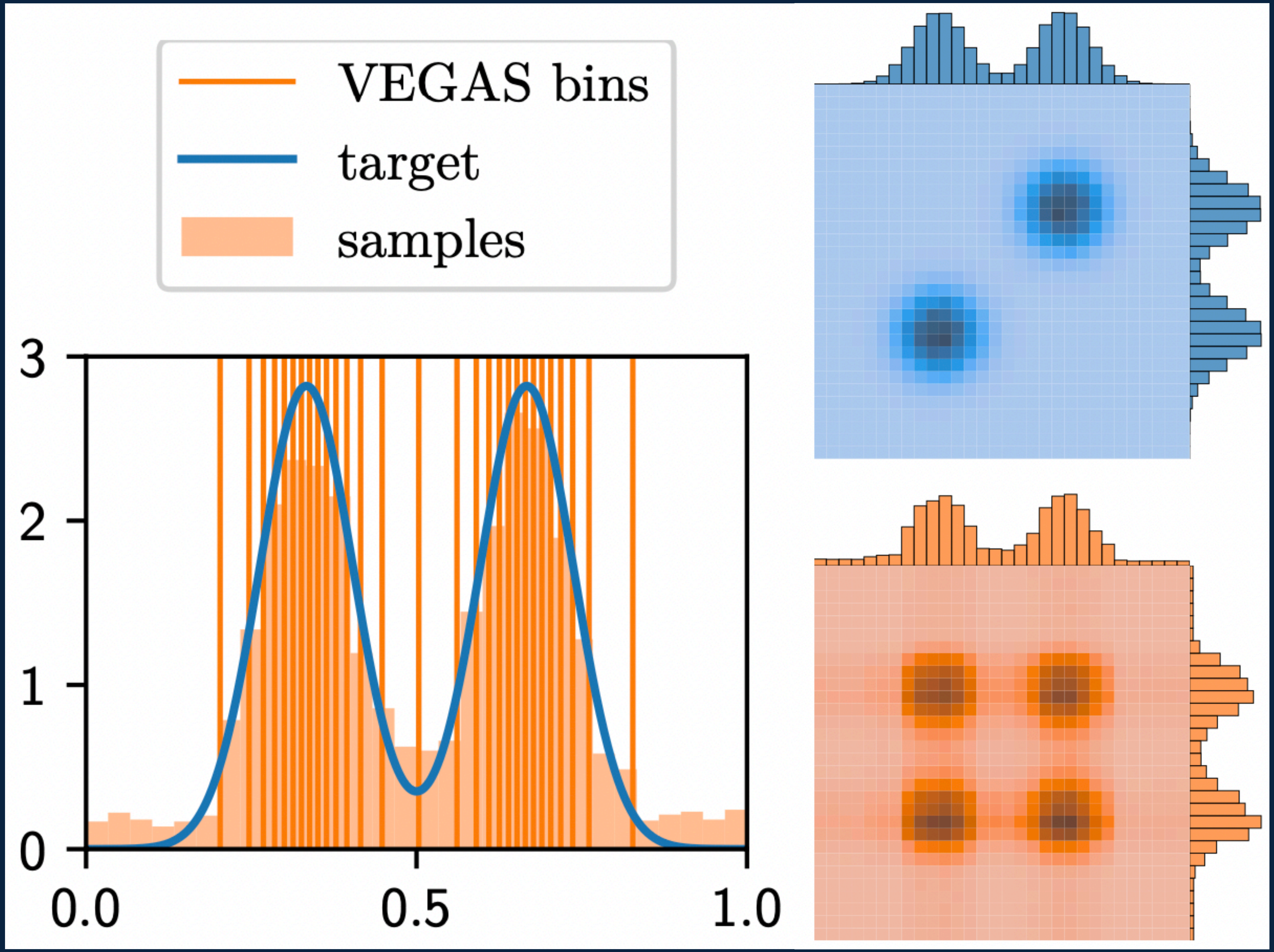
$$I = \left\langle \frac{f(x)}{g(x)} \right\rangle_{x \sim g(x)}$$



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



Why not VEGAS for everything?

- High-dim and rich peaking → slow convergence
- If peaks are not aligned with grid axes → "phantom peaks"

Importance sampling — NN

Using a Neural Network

- Unbinned and no grids
  → no “phantom peaks”
- Bijectivity not guaranteed
  → training unstable
- Numerical Jacobians
  → slow training and evaluation

[1707.00028, 1810.11509, 2009.07819]

Importance sampling — Flow

Using a Neural Network

- Unbinned and no grids
  → no “phantom peaks”
- Bijectivity not guaranteed
  → training unstable
- Numerical Jacobians
  → slow training and evaluation

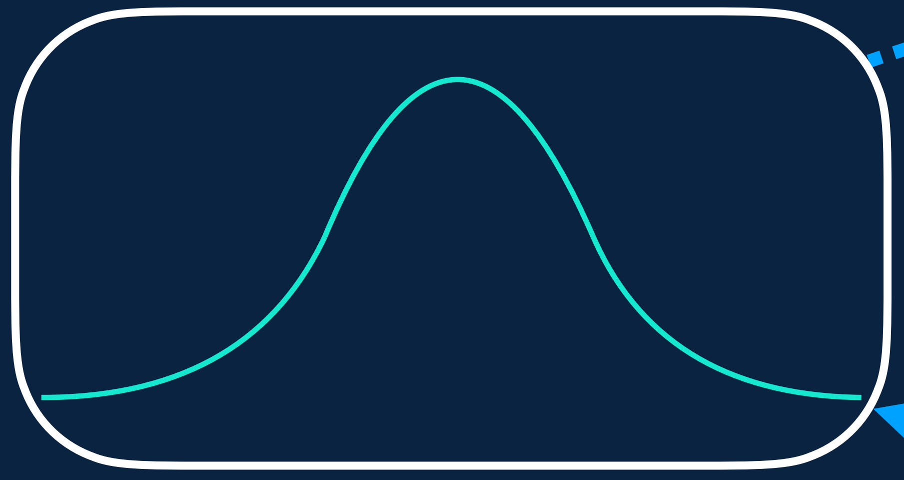
[1707.00028, 1810.11509, 2009.07819]

Using a Flow instead

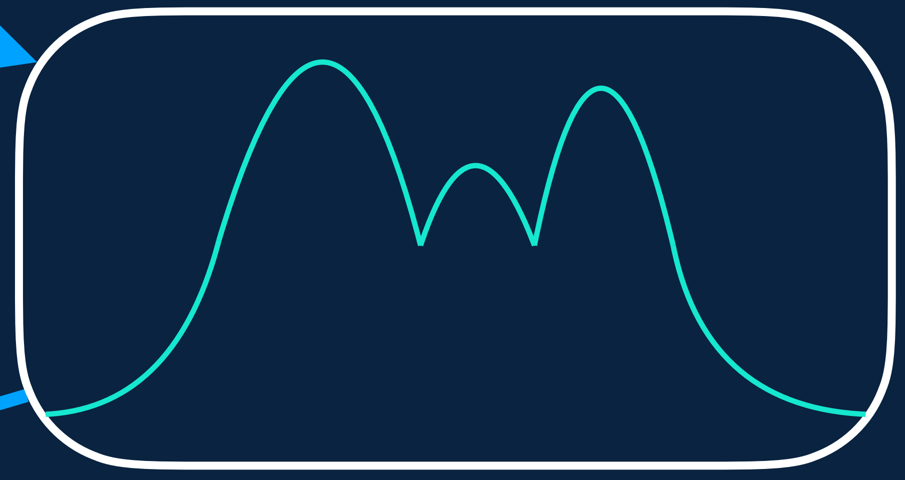
- Invertibility
  → bijective mapping
- tractable Jacobians
  → fast training and evaluation

[2001.05478, 2001.05486, 2001.10028, 2005.12719, 2112.09145]

Normalizing Flow



$$\log p_y(y) = \log p_x(x) + \log \left| \frac{\partial G(x)}{\partial x} \right|$$



MadNIS

Neural Importance Sampling

[2212.06172]

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

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

$\downarrow$

Use physics knowledge to construct channel 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 channel and mappings

Normalizing flow to refine channel mappings

Fully connected network to refine channel weights

MadNIS — Neural importance sampling

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

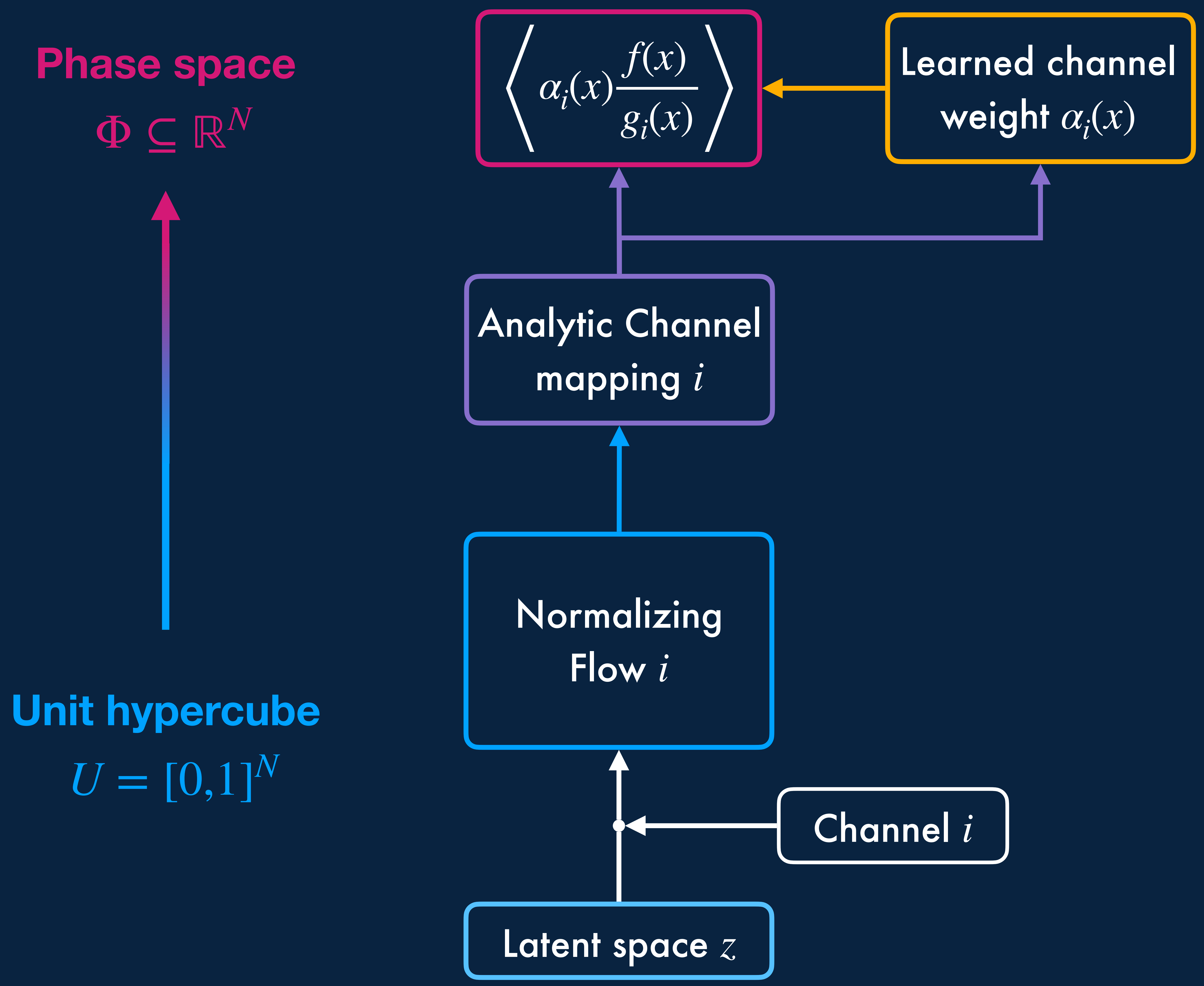
Use physics knowledge to construct channel and mappings

Normalizing flow to refine channel mappings

Fully connected network to refine channel weights

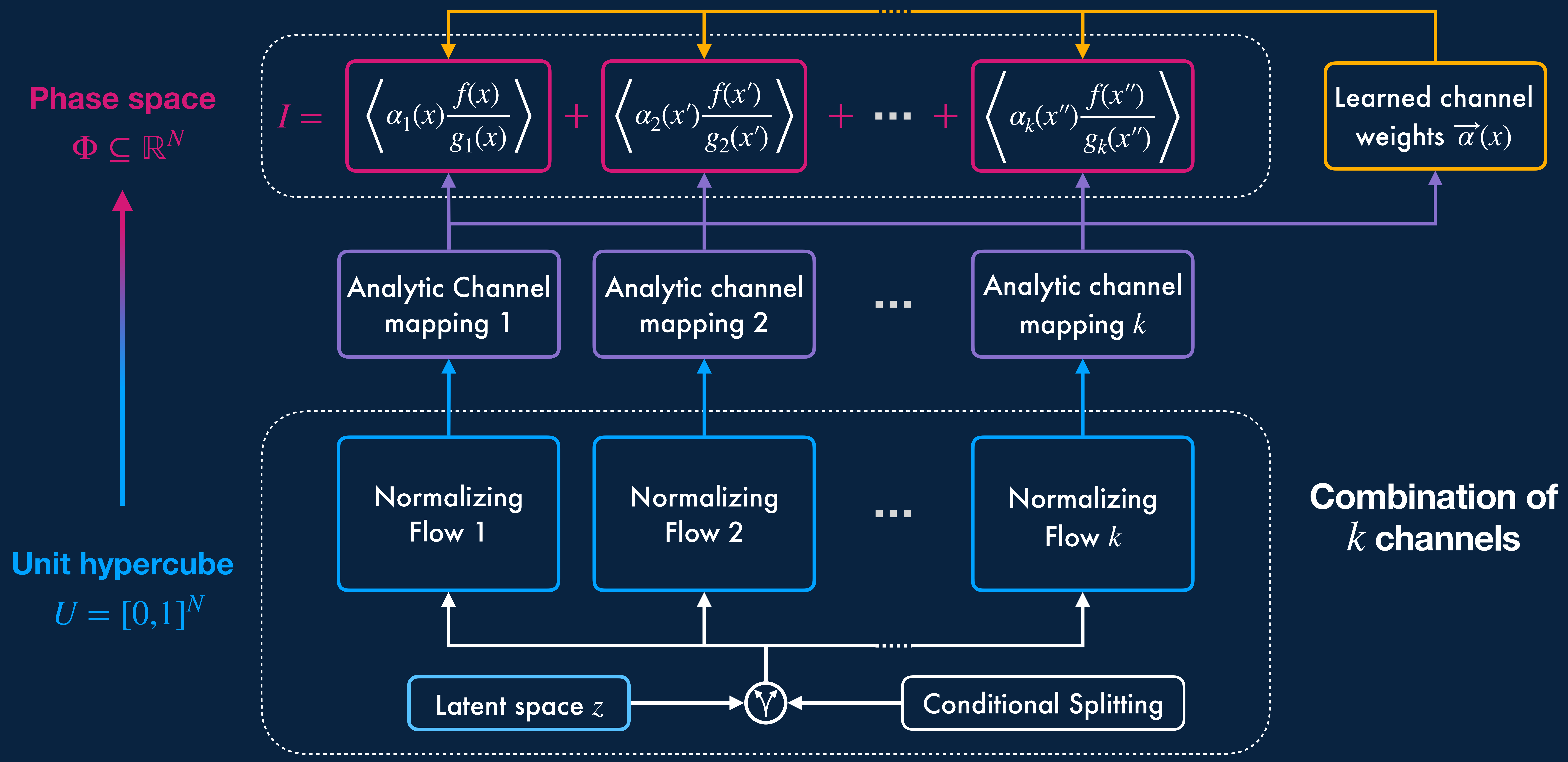
Update simultanously with variance as loss function

MadNIS — Neural importance sampling



Single channel i

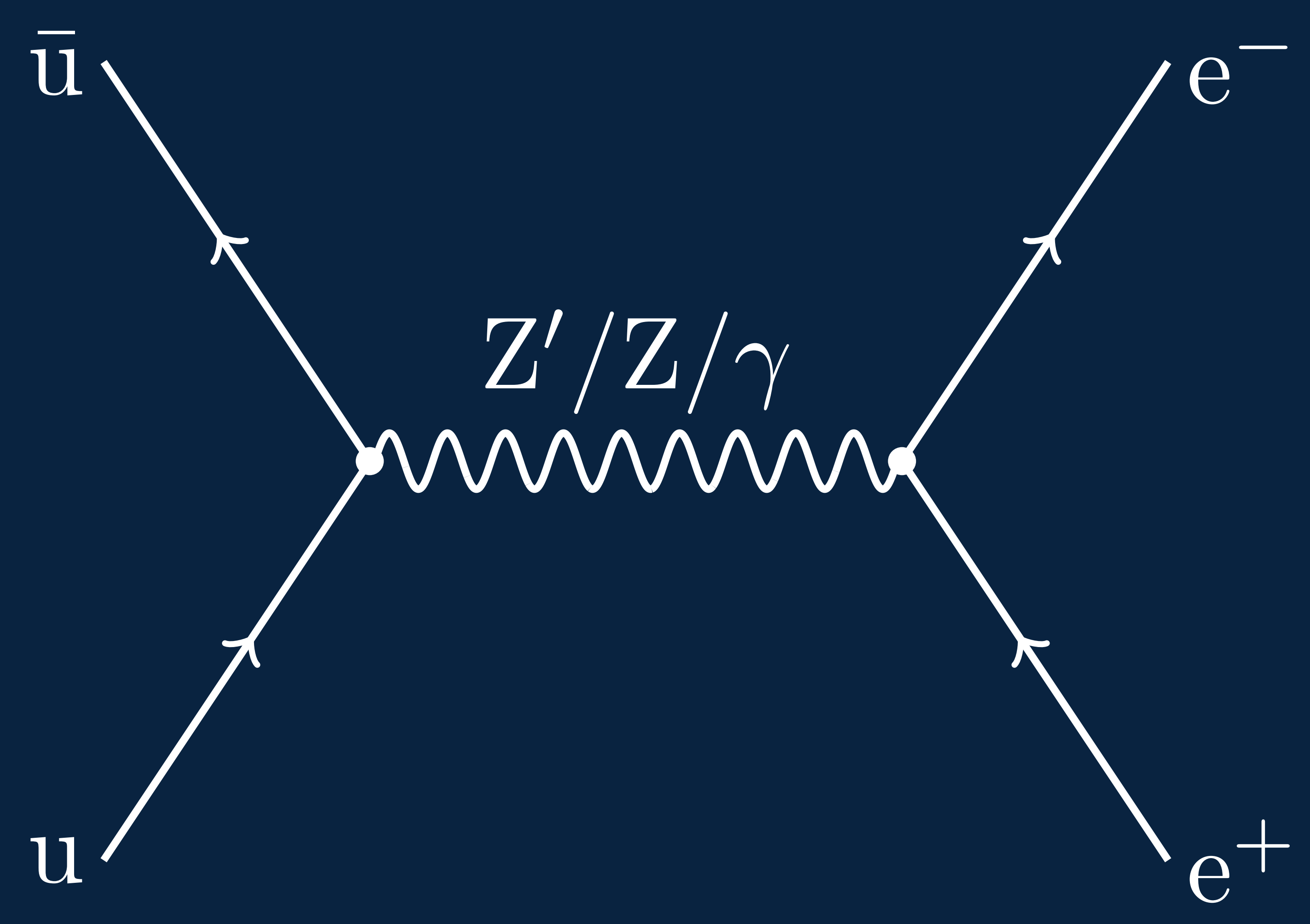
MadNIS — Neural importance sampling

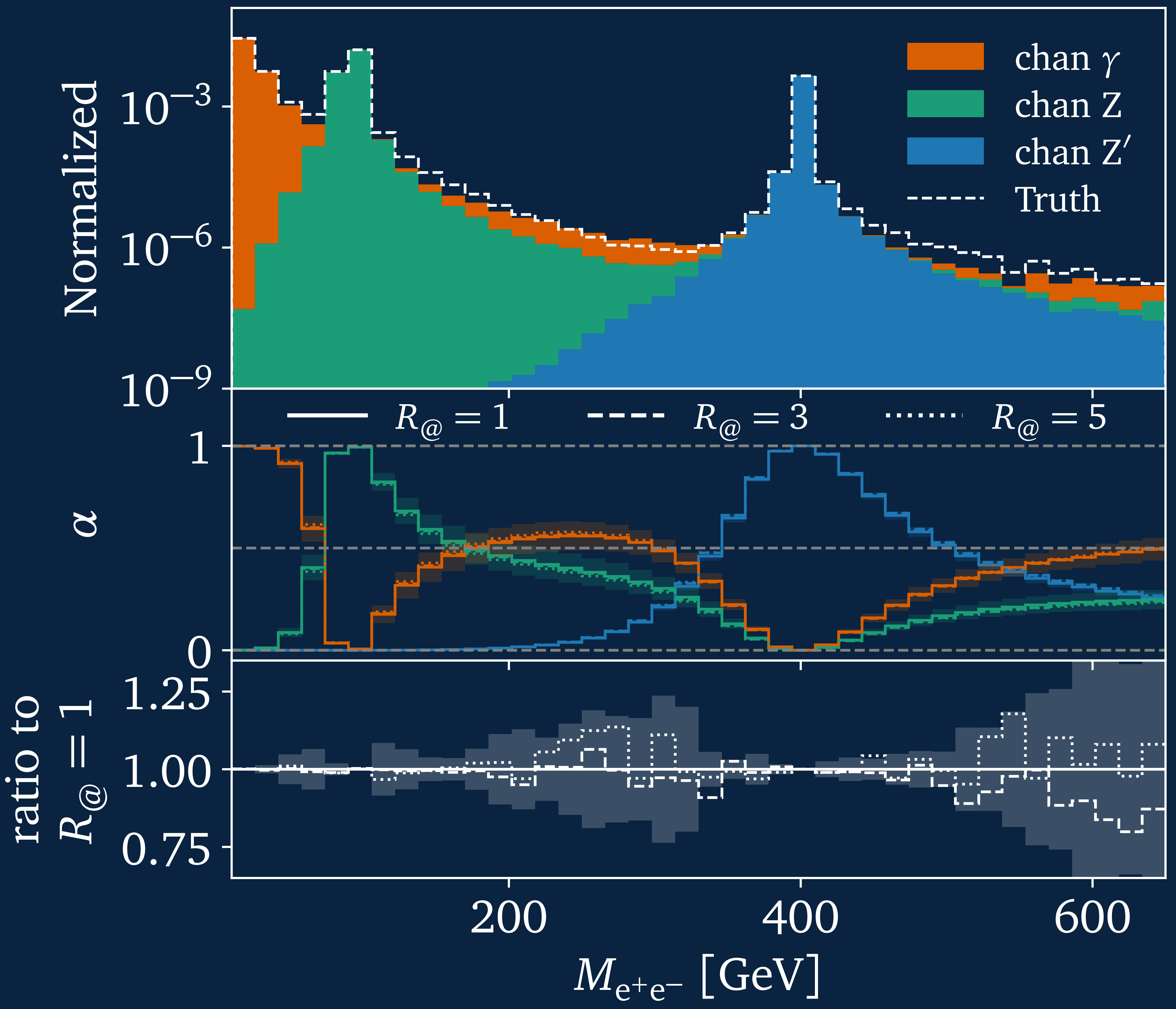


Toy Example — Drell-Yan + Z’

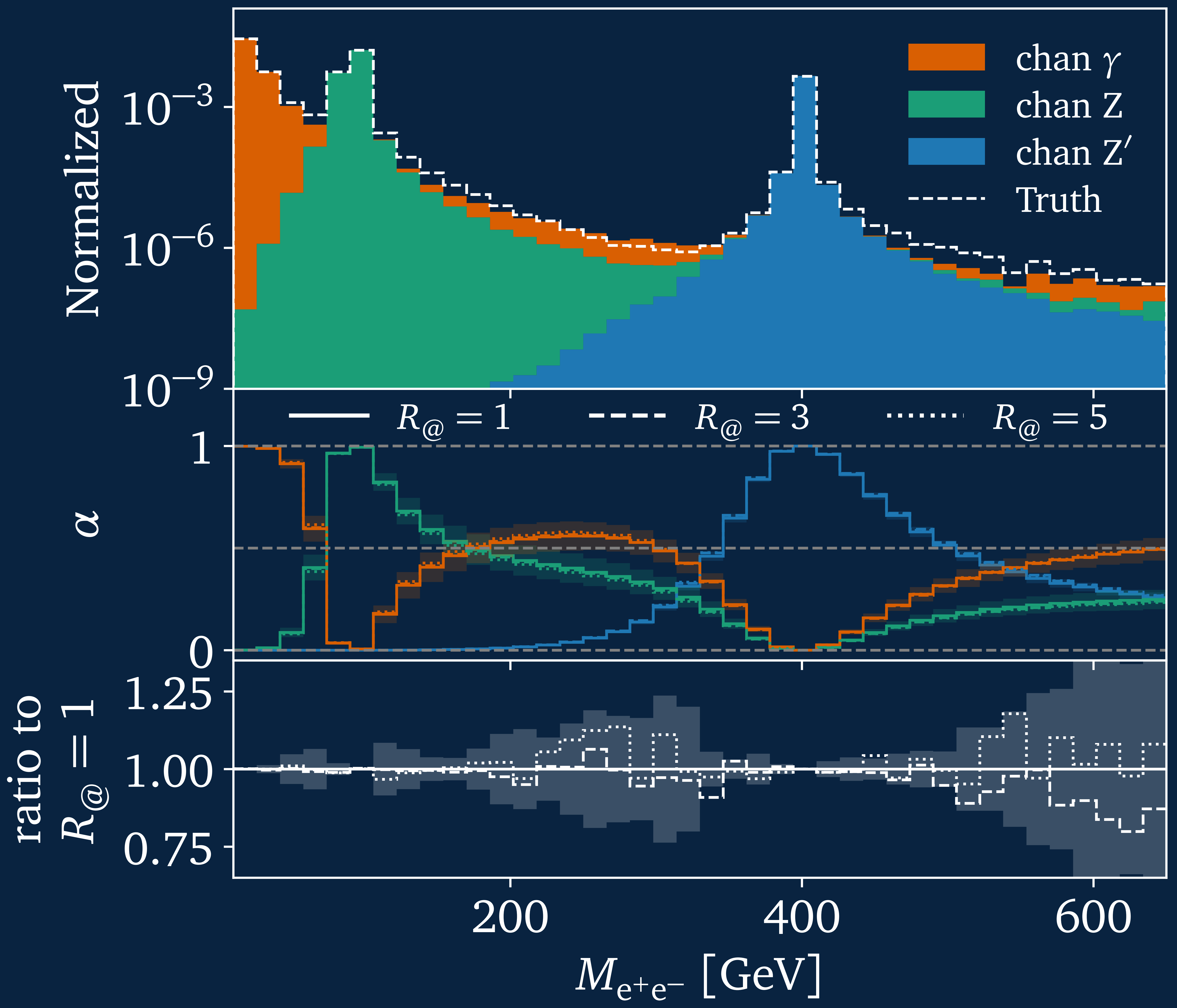
Implementation

- Custom amplitude in TensorFlow2
- Custom PS mappings in TensorFlow2
- PDFs from LHAPDF [1412.7420]



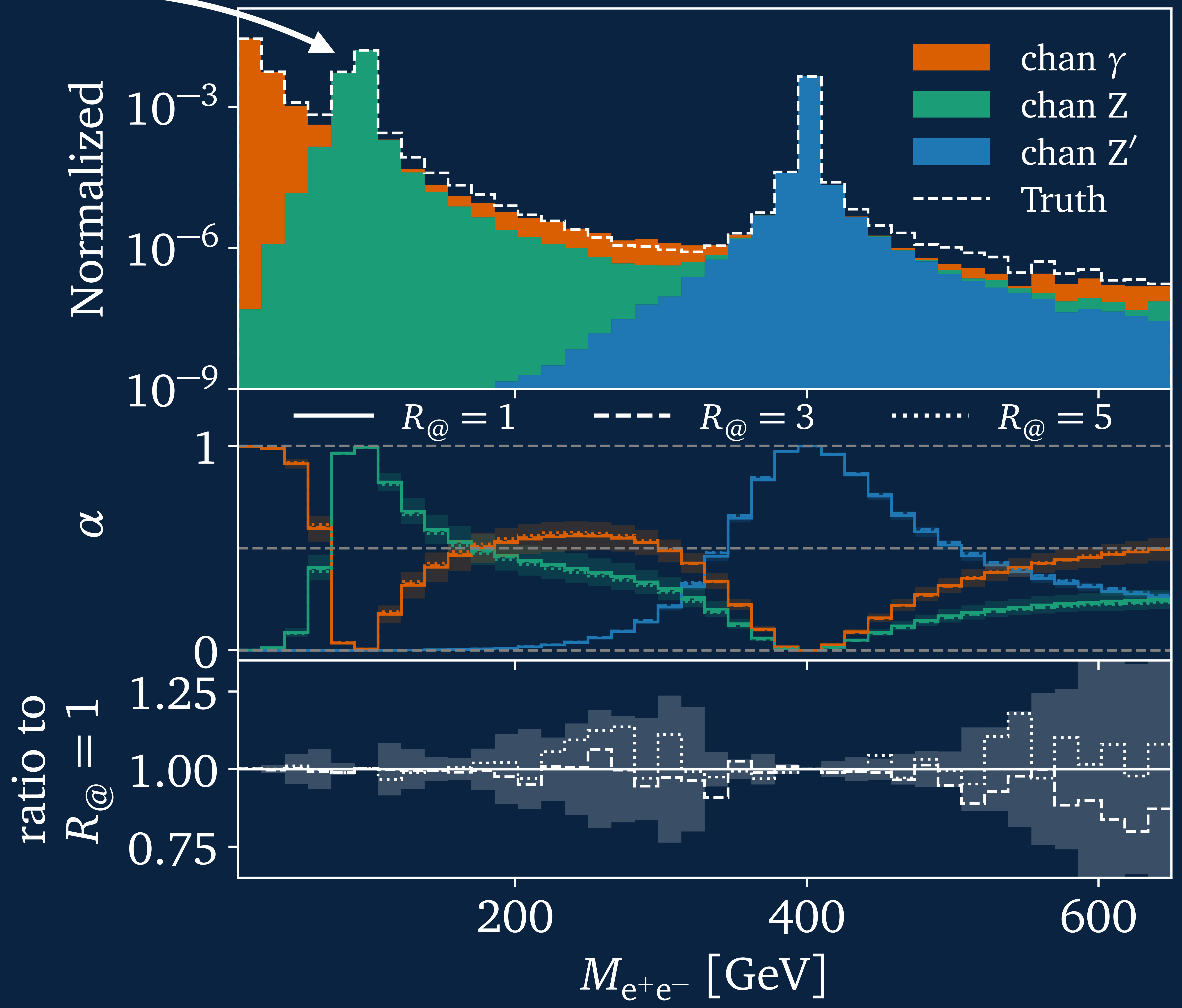


Toy Example — Results



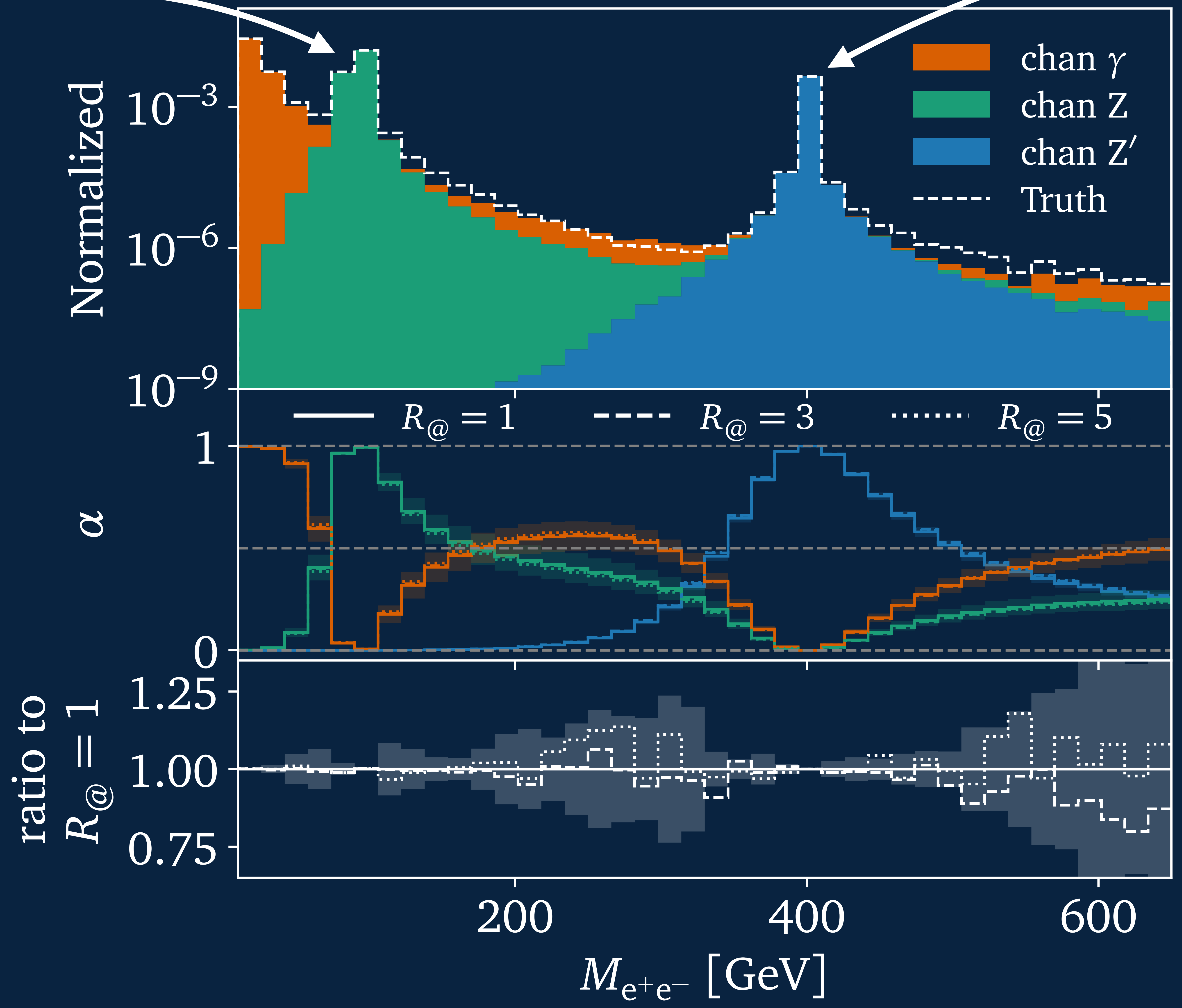
Toy Example — Results

Learned distribution matches truth



Toy Example — Results

Learned distribution matches truth

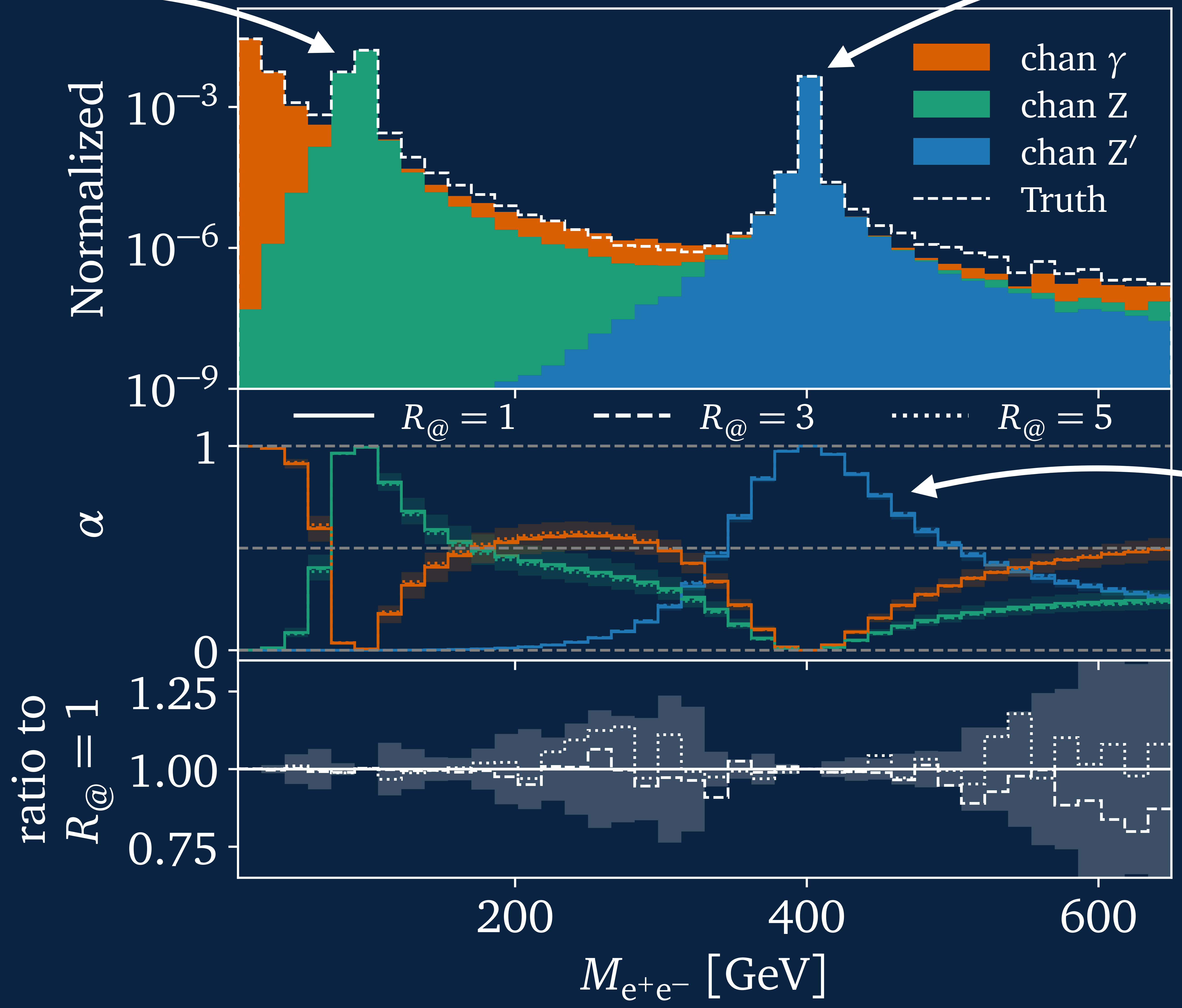


Peaks mapped out by different channels

Toy Example — Results

Learned distribution matches truth

Peaks mapped out by different channels

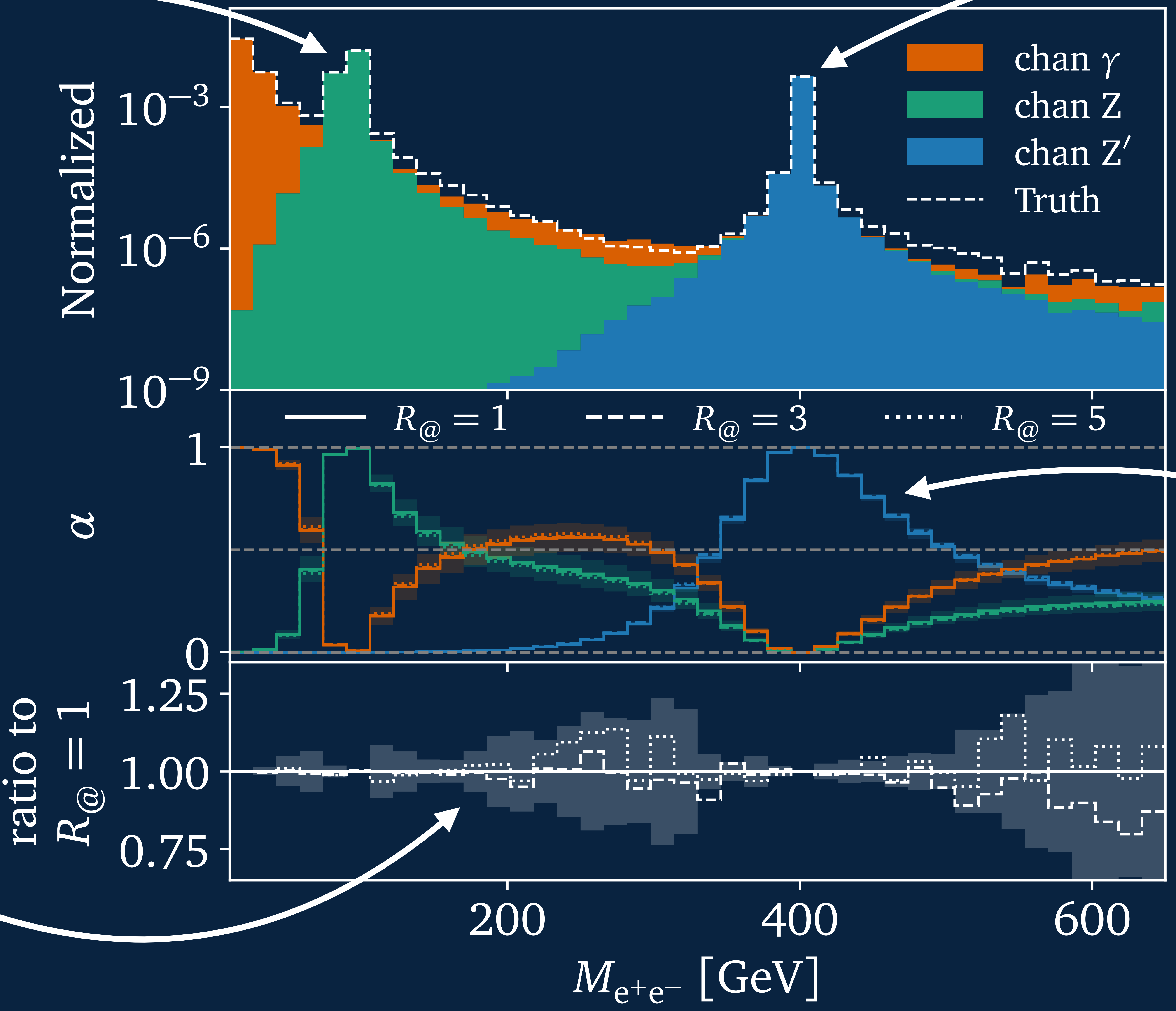


Channel weights learned by network

Toy Example — Results

Learned distribution matches truth

Use samples multiple times to make training faster



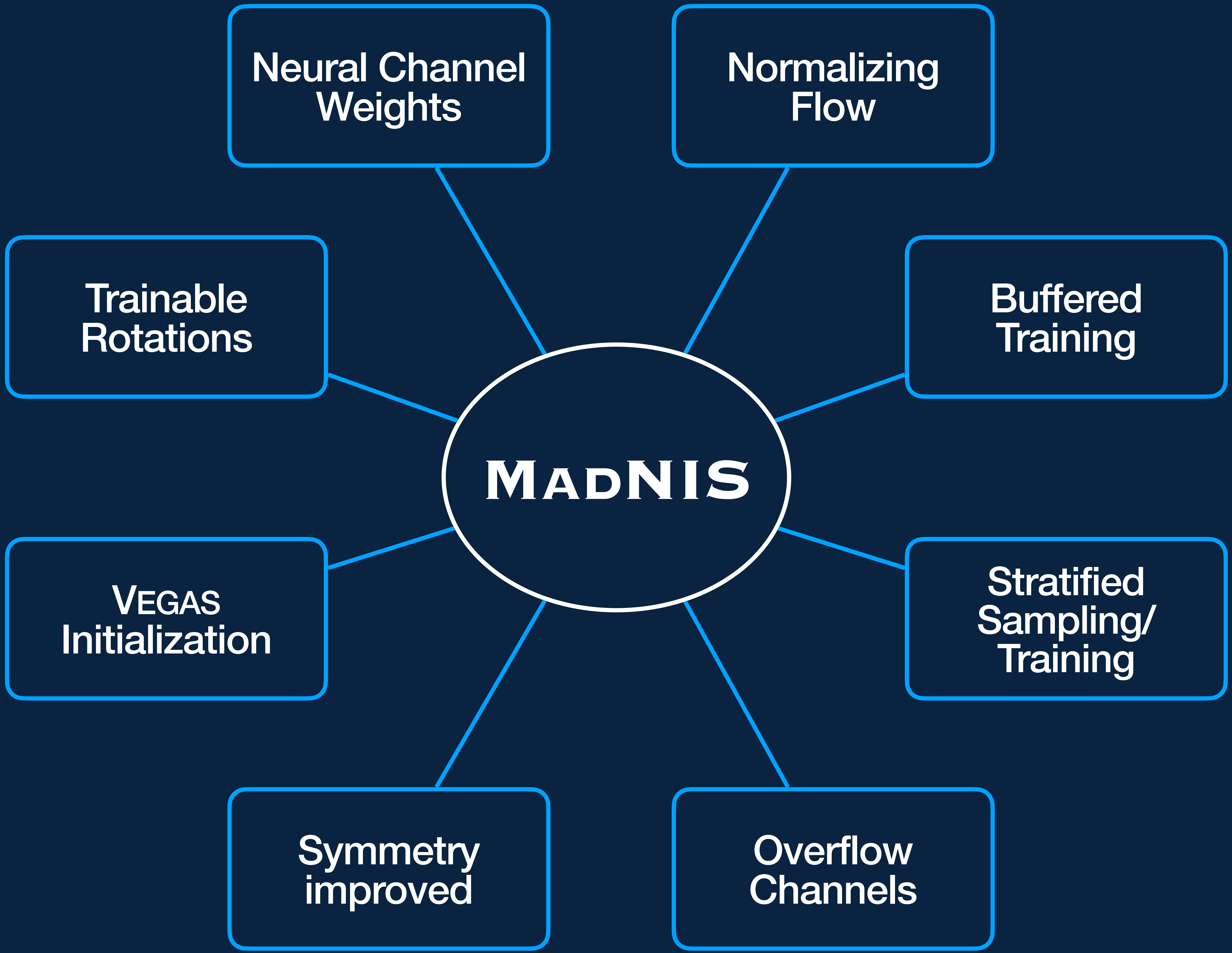
Peaks mapped out by different channels

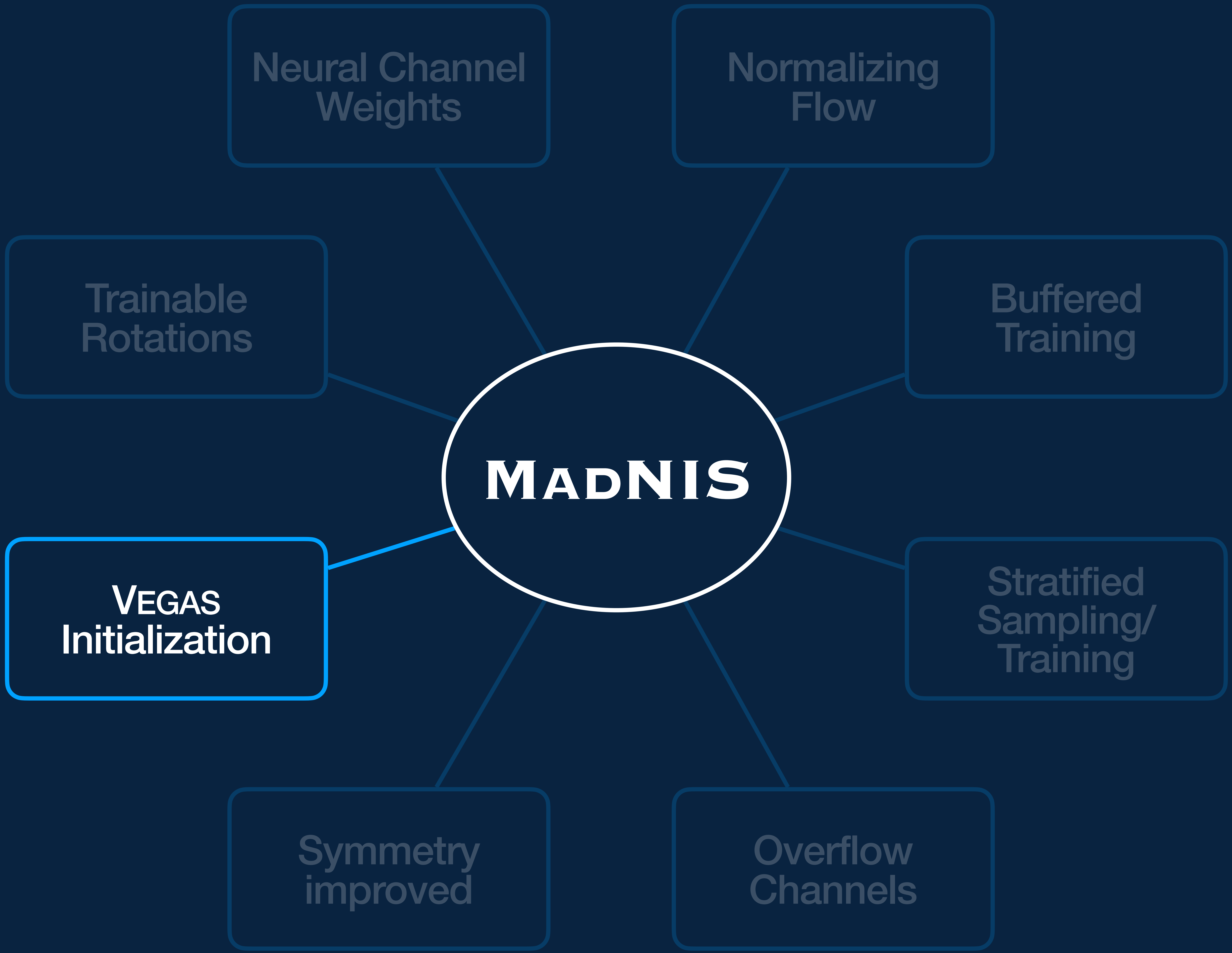
Channel weights learned by network

Can we beat standard frameworks?

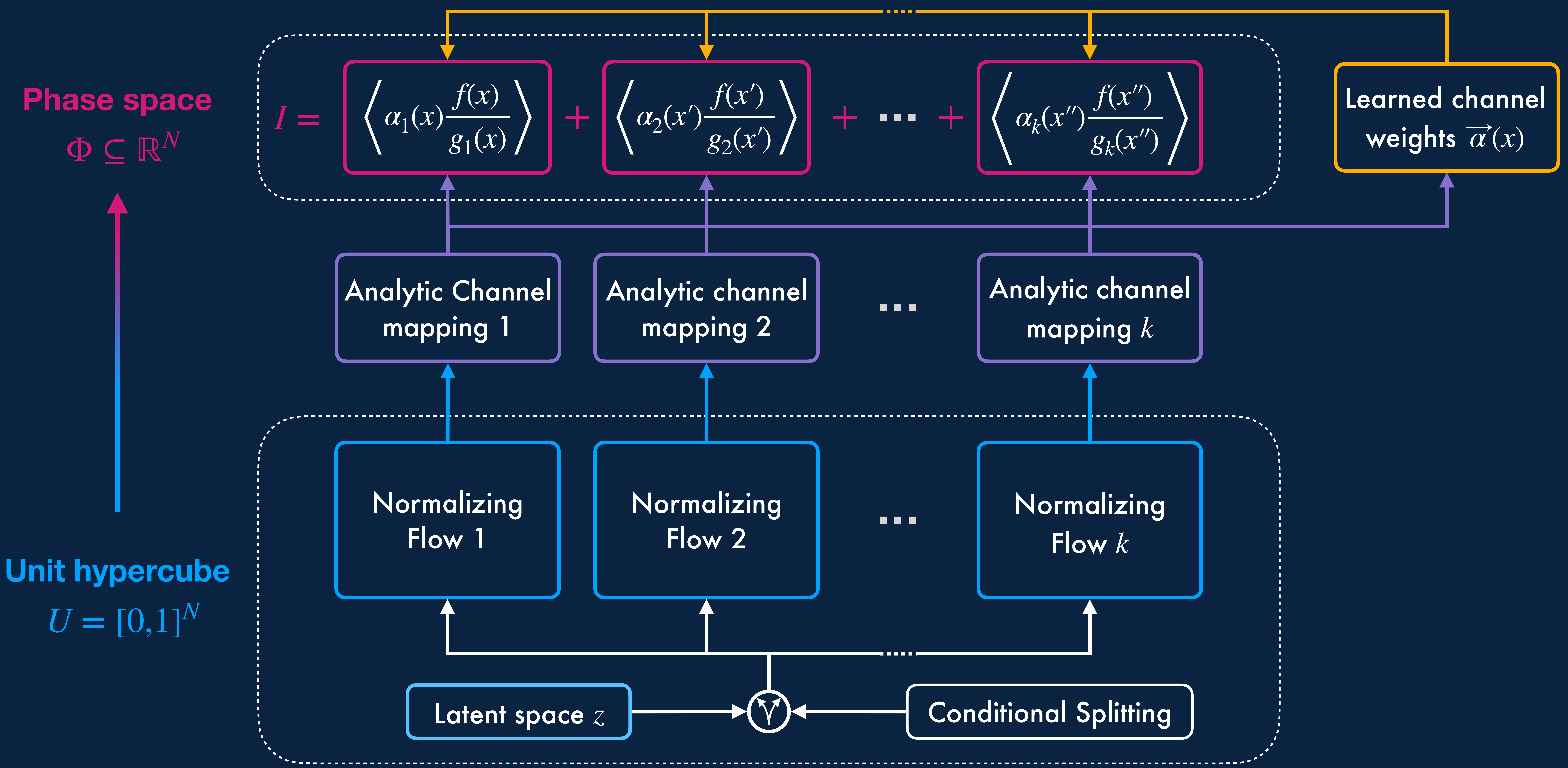
MadNIS Reloaded

How to beat MadGraph

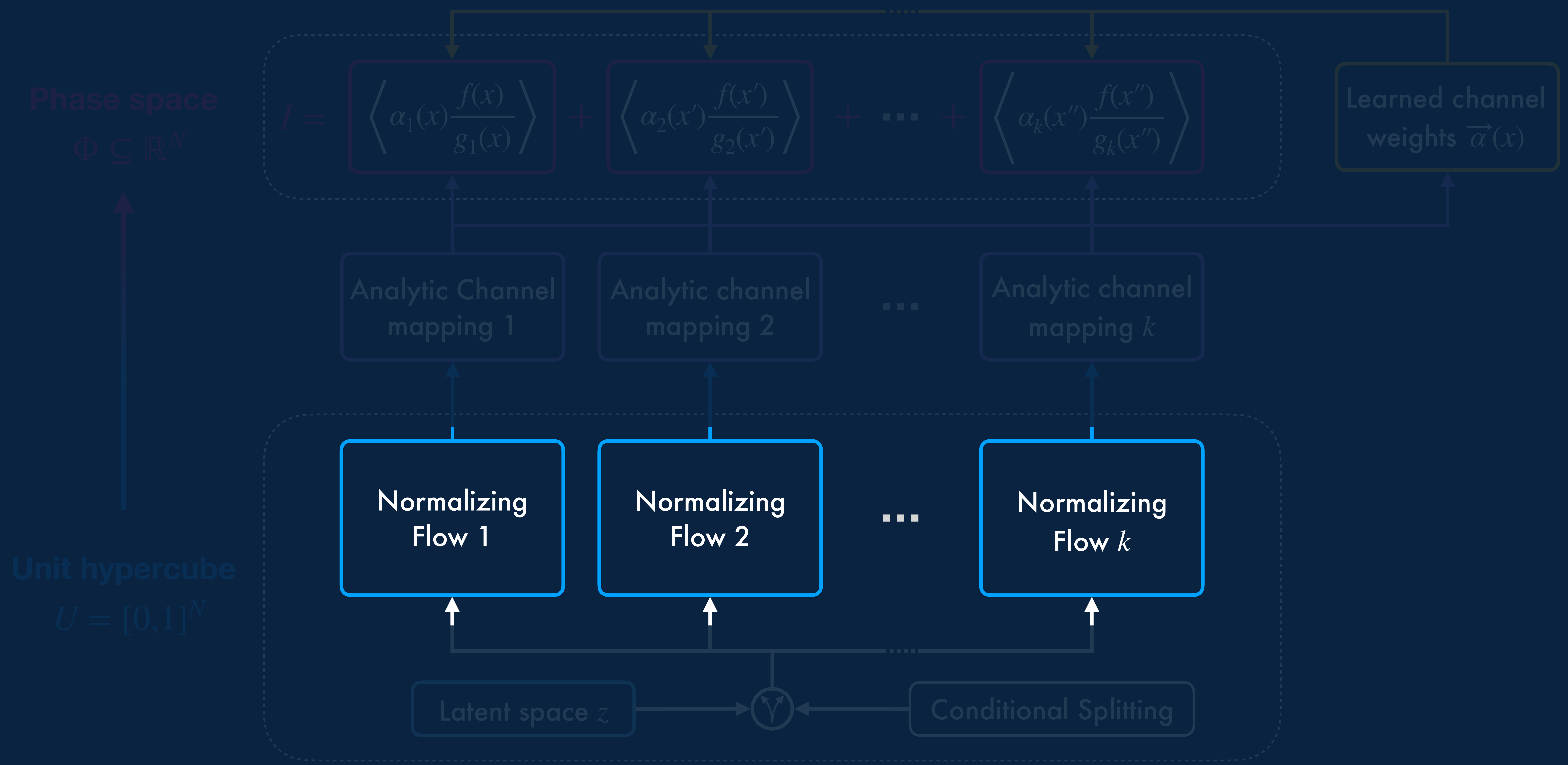

Neural Channel Weights
Normalizing Flow
Trainable Rotations
Buffered Training
MadNIS
VEGAS Initialization
Stratified Sampling/ Training
Symmetry improved
Overflow Channels


Neural Channel Weights
Normalizing Flow
Trainable Rotations
Buffered Training
MadNIS
VEGAS Initialization
Stratified Sampling/ Training
Symmetry improved
Overflow Channels

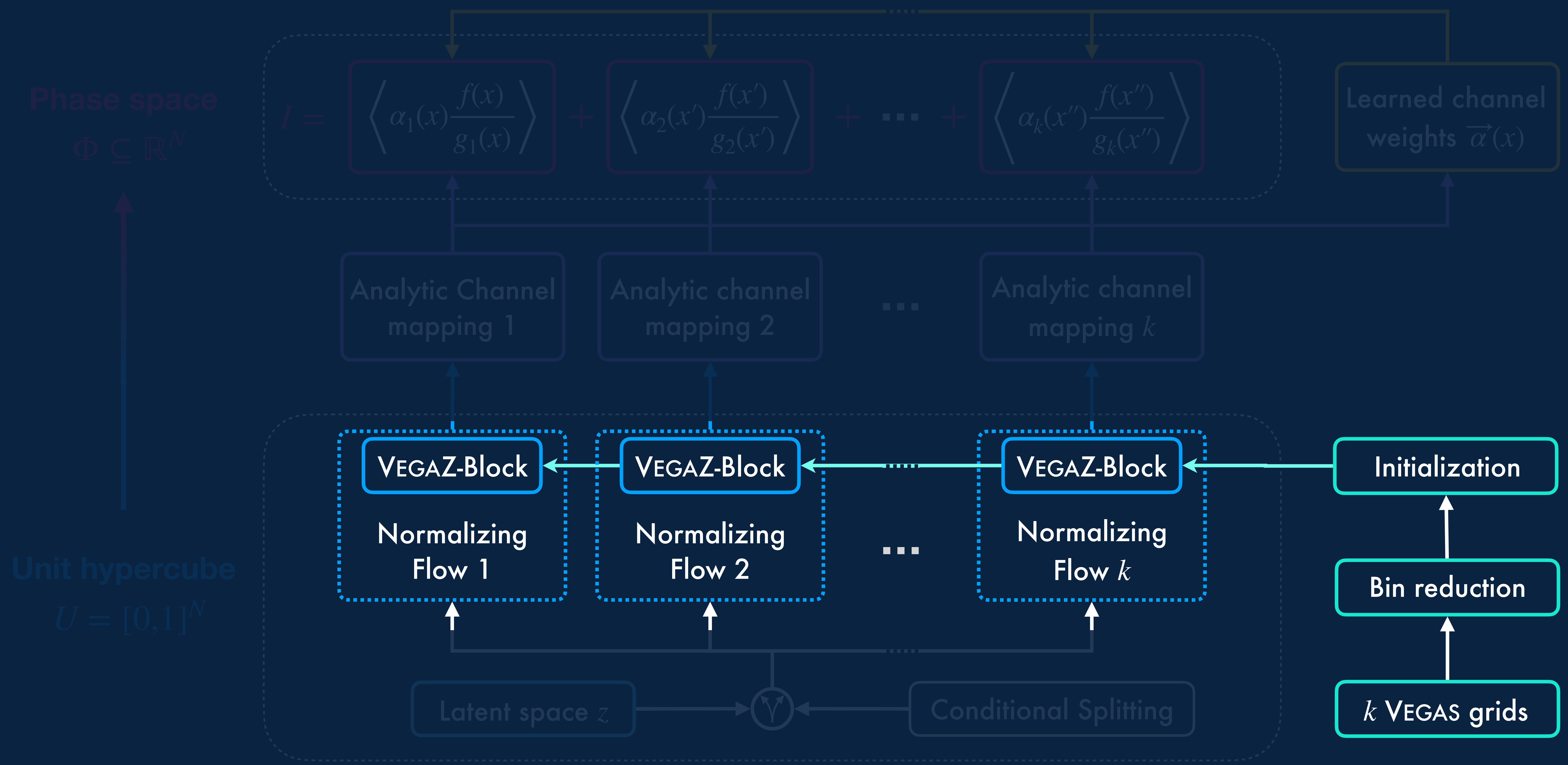
MadNIS — VEGAZ-Block



MadNIS — VEGAZ-Block



MadNIS — VEGAZ-Block



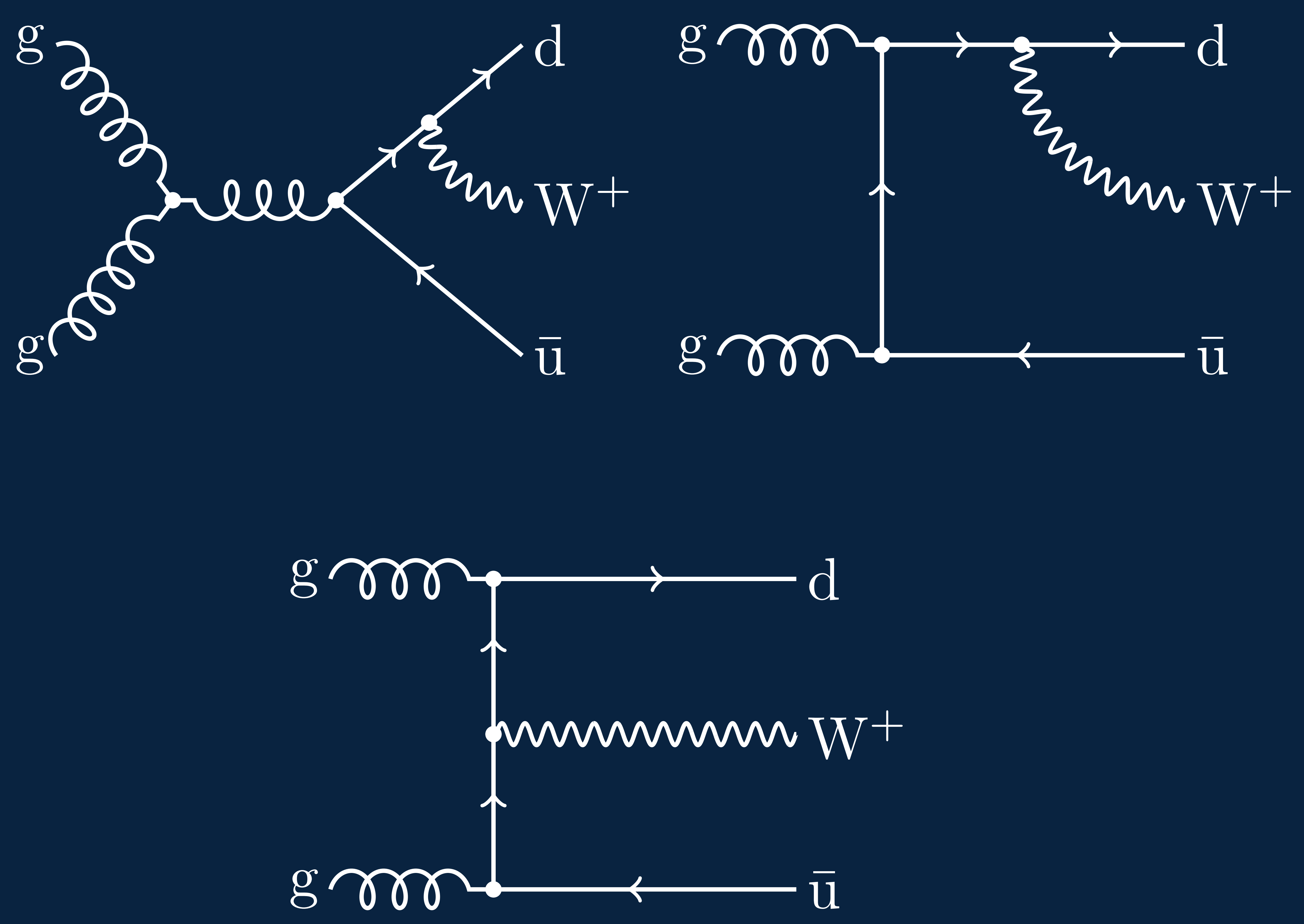
First benchmark — W+2jets

Implementation

- Amplitude and PS mapping from MadGraph
- Direct implementation via MadGraph-API

MadGraph API

- MadNIS can access and use (almost) all features of MadGraph
- Automatically generates necessary files for arbitrary processes (LO only)



First benchmark — Results

8 Channels	Integral [pb]	Relative stddev	Unweighting eff.
MG5AMC*	216.4(8)	2.13	2.3%
Flow	215.20(14)	0.64	9.0%
VegaZ-Flow	215.13(12)	0.57	11.1%
α-VEGAZ-Flow	**215.07(11)**	**0.55**	**11.7%**

First benchmark — Results

8 Channels	Integral [pb]	Relative stddev	Unweighting eff.
MG5AMC*	216.4(8)	2.13	2.3%
Flow	215.20(14)	0.64	9.0%
VegaZ-Flow	215.13(12)	0.57	11.1%
α-VEGAZ-Flow	**215.07(11)**	**0.55**	**11.7%**

4 Channels	Integral [pb]	Relative stddev	Unweighting eff.
MG5AMC*	215.4(4)	1.39	3.9%
Flow	215.10(11)	0.53	14.2%
VegaZ-Flow	214.96(11)	0.49	14.8%
α-VEGAZ-Flow	**215.00(10)**	**0.47**	**15.5%**

What is the future of MadGraph?

ML for MadGraph5_aMC@NLO

ML for MadGraph5_aMC@NLO + Future

Summary and Outlook

Summary

- MadNIS outperforms current sampling methods
- Multi-channel is more efficient when trained simultanously with the flow
- Vegas initialization improves performance

Outlook

- Fully integrate MadNIS into MadGraph
- Test performance on real LHC examples: (eg. multi-leg, NLO, complicated cuts, …)
- Make everything run on the GPU and differentiable [MadJax 2203.00057]

Summary and Outlook

HEPML-LivingReview

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

The purpose of this note is to collect references for modern machine learning as applied to particle physics. A minimal number of categories is chosen in order to be as useful as possible. Note that papers may be referenced in more than one category. The fact that a paper is listed in this document does not endorse or validate its content - that is for the community (and for peer-review) to decide. Furthermore, the classification here is a best attempt and may have flaws - please let us know if (a) we have missed a paper you think should be included, (b) a paper has been misclassified, or (c) a citation for a paper is not correct or if the journal information is now available. In order to be as useful as possible, this document will continue to evolve so please check back before you write your next paper. If you find this review helpful, please consider citing it using \cite{hepmllivingreview} in HEPML.bib.

Outlook

- Fully integrate MadNIS into MadGraph
- Test performance on real LHC examples:
  (eg. multi-leg, NLO, complicated cuts, …)
- Make everything run on the GPU and differentiable [MadJax 2203.00057]
- Stay tuned for many other ML4HEP applications

HEPML